Generating computer forensic super-timelines under Linux

A comprehensive guide for Windows-based disk images

R. Carbone
Certified Hacking Forensic Investigator (EC Council)
DRDC Valcartier

C. Bean
Certified Hacking Forensic Investigator (EC Council)

Defence R&D Canada – Valcartier

Technical Memorandum
DRDC Valcartier TM 2011-216
October 2011

Principal Author

Original signed by Richard Carbone

Richard Carbone

Programmer/Analyst

Approved by

Original signed by Guy Turcotte

Guy Turcotte

Head/System of Systems

Approved for release by

Original signed by Christian Carrier

Christian Carrier

Chief Scientist

Abstract

This technical memorandum examines the basics surrounding computer forensic filesystem timelines and provides an enhanced approach to generating superior timelines for improved filesystem analysis and contextual awareness. Timelines are improved by polling multiple sources of information across the filesystem resulting in an approach that is surprisingly flexible and customizable. The timeline is further enhanced by incorporating key time-based metadata found across a disk image which, when taken as a whole, increases the forensic investigator's understanding.

Résumé

Ce mémorandum technique examine les bases entourant la création d'un calendrier des événements inforensiques des systèmes de fichier et fournit une approche améliorée pour générer des calendriers supérieurs pour une analyse améliorée des systèmes de fichiers et un meilleur éveil contextuel. Ces calendriers sont améliorés en sondant des sources multiples d'information à travers le système de fichiers, ce qui résulte en une approche qui est étonnamment flexible et configurable. Le calendrier est amélioré encore davantage par l'introduction des métadonnées essentielles liées au temps qui se retrouvent un peu partout sur un disque et qui, lorsque prises en compte globalement, augmentent la compréhension de l'enquêteur inforensique.

This page intentionally left blank.

Executive summary

Generating computer forensic super-timelines under Linux: A comprehensive guide for Windows-based disk images

Carbone, R., Bean, C.; DRDC Valcartier TM 2011-216; Defence R&D Canada – Valcartier; October 2011.

Modern digital forensics relies on a multitude of software tools and investigative techniques in an attempt to understand and piece together the actions taken by a suspect. Many pieces of relevant information exist, including precious metadata. It is this metadata, specifically date/time related metadata, which constitutes the main subject of this technical memorandum. Unfortunately, the full utilization of said metadata is all too often left out from an investigation as only the basic filesystem date/time metadata is actually used. Using additional metadata sources, as examined in this technical memorandum, would further increase the confidence reliability of a digital forensic timeline.

These timelines, while in use for many years now by investigators, are lacking in context and depth. Far too many important additional sources of date/time-related metadata are left out from the timeline, thereby limiting the investigator's contextual understanding of events. By increasing the scope and range of additional metadata and pushing it into the timeline, new insights can be gained, particularly into user-related activities.

While these concepts are not new, they are not widely used today by investigators nor are they implemented into current commercial digital forensic analysis tools and frameworks. Instead, digital forensic timelines appear to have been ignored by the software industry. However, there has been some resurgence of interest in timelines and the software tools at the forefront, *log2timeline* and The Sleuth Kit, play a key role in the timeline generation approach described herein.

Specifically, the open source tool *log2timeline*, in combination with custom shell scripts and short C-based programs, will enable the reader to not only generate enhanced timelines from additional sources of date/time-related metadata but allow for said generation in a more automatable fashion. This technical memorandum therefore provides a detailed description of the authors' approach to generating enhanced timelines in the hopes of aiding the digital forensic community to adopt improved methodologies.

This work was carried out over a period of two months as part of the Live Computer Forensics project, an agreement between DRDC Valcartier and the RCMP (SRE-09-015, 31XF20). The results of this project will be of great interest to the Canadian Forces Network Operation (CFNOC) in their mission of securing DND networks and investigating computer incidents.

Sommaire

Generating computer forensic super-timelines under Linux: A comprehensive guide for Windows-based disk images

Carbone, R., Bean, C. ; DRDC Valcartier TM 2011-216 ; R & D pour la défense Canada – Valcartier; octobre 2011.

L'inforensique moderne repose sur une multitude d'outils logiciels et de techniques d'enquête dans le but de comprendre et de réunir les actes commis par un suspect. Beaucoup d'information pertinente existe, incluant de précieuses métadonnées. Ce sont ces métadonnées, partiellement celles liées à la date et à l'heure, qui constituent le sujet principal de ce mémorandum technique. Malheureusement, la pleine utilisation de ces métadonnées est trop souvent mise de côté dans une enquête et seules les métadonnées de base du système de fichiers, liées à la date et à l'heure, sont utilisées dans les faits. L'utilisation de sources de métadonnées additionnelles, telles qu'examinées dans ce mémorandum technique, augmenterait le niveau de fiabilité envers un calendrier des événements inforensiques.

Ces calendriers, pourtant utilisés depuis plusieurs années par les enquêteurs, manquent d'information contextuelle et de profondeur. Beaucoup trop de métadonnées additionnelles importantes liées à la date et à l'heure sont mise à l'écart du calendrier, ce qui limite la compréhension contextuelle des événements pour l'enquêteur. En augmentant la portée et la gamme des métadonnées additionnelles et en les incluant dans le calendrier des événements, de nouvelles connaissances peuvent être acquises, particulièrement en ce qui a trait aux activités des utilisateurs.

Bien que ces concepts ne soient pas nouveaux, ils sont peu utilisés par les enquêteurs et ils ne sont pas implémentés dans les outils et cadres courants d'inforensique commerciaux. Les calendriers des événements numériques semblent plutôt avoir été négligés par l'industrie logicielle. Malgré tout, il semble y avoir eu un regain d'intérêt pour ces calendriers et les outils logiciel à l'avant-garde, *log2timeline* et The Sleuth Kit, joue un rôle clé dans la production de calendriers des événements décrits dans ce mémorandum.

Spécifiquement, l'outil en logiciel libre *log2timeline*, combiné avec des scripts de commandes personnalisés et de petits programmes en C, permettra au lecteur non seulement de produire des calendriers améliorés à partir de sources additionnelles de métadonnées liées à la date et à l'heure, mais permettra aussi une meilleure automatisation de cette production. Ce mémorandum technique fournit donc une description détaillée de l'approche des auteurs pour produire des calendriers des événements plus détaillés dans l'espoir d'aider la communauté inforensique à adopter des méthodes améliorées.

Ce travail a été accompli sur une période de deux mois dans le cadre du projet « Live Computer Forensics », une entente entre RDDC Valcartier et la GRC (SRE-09-015, 31XF20). Les résultats de ce projet seront d'un grand intérêt pour le Centre d'opérations des réseaux des Forces canadiennes (CORFC) dans leur mission de protection des réseaux du MDN et d'investigation des incidents informatiques.

This page intentionally left blank.

Table of contents

List of tables

Acknowledgements

The authors would like to thank Mr. Yves van Chestein for peer-reviewing this document and providing many useful comments and insight in order to improve it and to Mr. Martin Salois for translating select portions of this text.

Disclaimer

The reader should neither construe nor interpret the work described herein by the authors as an endorsement of the aforementioned techniques and capacities as suitable for any specific purpose, construed, implied or otherwise.

Furthermore, the aforementioned authors absolve themselves in all ways conceivable with respect to how the reader may use, interpret or construe this technical memorandum. The authors assume absolutely no liability or responsibility, implied or explicit. Moreover, the onus is on the reader to be properly equipped and knowledgeable in the application of digital forensics.

Finally, the authors, the Government of Canada, the Minister of National Defence (Canada), the Department of National Defence (Canada) and Defence Research and Development Canada are henceforth absolved of all wrongdoing, whether intentional, unintentional, construed or misunderstood on the part of the reader. If the reader does not agree to these terms, then this technical memorandum should be readily returned to the Department of National Defence (Canada). Only if the reader agrees to these terms should he or she continue reading it beyond this point. It is further assumed by all participants that the reader has read this Disclaimer and assumes all responsibility for any repercussions which may result from the information and data contained herein.

C source code and Bash shell script code disclosure licensing agreement

All source code included herein is hereby released to the community at large using a BSD-based licensing agreement. The actual license by which all code is released to the public is as follows:

C code and Bash shell scripts presented herein were written and coded by Richard Carbone, Defence R&D Canada – Valcartier, 2011.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Re-distributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Re-distributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the authors, Defence R&D Canada, or the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Requirements and assumptions

It is assumed that the reader is altogether familiar with digital forensics and the various techniques and methodologies associated therein. This technical memorandum is neither an introduction to digital forensics or said techniques and methodologies. However, this technical memorandum will endeavour to present sufficient technically-oriented background information so that the reader can understand why additional sources of date/time-based metadata are required, the consequences of timeline usage and generation and enable to him to modify the work presented herein for his own specific needs.

The present technical memorandum examines only timeline generation for forensically acquired disk images. Timeline generation from live systems is not examined herein.

All work presented herein has been carried out using a Linux-based operating system and it is very likely that the reader's success in recreating similar results using the presented techniques and methodologies as they pertain to digital forensic timelines will require the use of such an operating system.

The reader should be both familiar and comfortable with shell scripting as it pertains to the Linux Bash shell. All shell scripts can of course be rewritten by the reader in another language including but not limited Perl, Python, etc. All C code was compiled using GCC version 4.4.5 20101112 under Fedora Core 13 64-bit with kernel 2.6.34.7-63.fc13.x86_64 #1 SMP. All Bash shell scripts were executed under Bash version 4.1.7(1)-release (x86_64-redhat-linux-gnu).

All Linux-based filesystem related experiments were conducted under Fedora Core 13 64-bit with kernel 2.6.34.7-63.fc13.x86_64 #1 SMP.

Software requirements

This technical report runs entirely atop Linux and requires The Sleuth Kit. Although it is possible to run the various programs including The Sleuth Kit, various Perl scripts and programs atop Windows the authors have made no effort to do so and leave it to reader to attempt.

The reader will require a C compiler such as GCC to compile the various C programs included in this report. The timeline control script was written to work atop Bash although it may run with other UNIX or Linux command line interpreters. Perl must be installed in order to run *log2timeline* and other scripts including their various dependencies which the reader must ensure are present in order for *log2timeline* and the scripts to function correctly. Harlan Carvey's Perl *regtime.pl* tool is also required by the timeline control script (see Annex B.1 for more details) and without it registry timeline extraction is not possible as *log2timeline*'s registry timeline extraction capability is somewhat more limited.

Standard UNIX processing tools including *sed*, *grep*, *tr*, *uniq*, *awk*, etc are generally included with most Linux distributions. If used under Windows, equivalents must be found and used.

Reasons why the timescanner tool was not used

When the authors began working on this technical memorandum in mid-2010 the *timescanner* tool was not yet released. Then, nearing the end of June 2010 the tool was released with version 0.50 of *log2timeline*. The tool did not work as anticipated and as such the authors continued with their work including their script and C programs (see annexes B.1, B.2.1, B.2.2 and B.2.3) rather than concentrate on a semi-functional tool.

As of mid-2011 the *timescanner* tool was working well and using it no longer required the use of complex or excessively long scripts, as was done for this work. However, in order for *timescanner* to work, many diverse Perl libraries must be installed which is not necessarily an obvious task to accomplish, even for experienced system administrators, as Perl library installations are at times known to be particular.

While the *timescanner* tool can be used to automatically find and extract date/time metadata using various prebuilt configuration files, this does not in any way negate the usefulness of *log2timeline*. In fact, quite the opposite is true. Although the investigator is often after efficient and automated means of extracting evidence from disk images, an overreliance on automation has the potential to significantly handicap an investigator. Specifically, when automation fails to work, a manual approach is required and this work, along with the technical background it provides and the variously aforementioned script and C programs fully address this.

By the time the *timescanner* tool became stable enough for operational use the vast majority of this work had already been completed and altogether abandoning it would mean that the comprehensive information it contains would not be readily shared with the digital forensics community. What's more, the ability to customise output, as is done in this work using the author-provided prototype control script, becomes an afterthought when using *timescanner* rather than having a highly useful timeline format produced as it is being generated.

Furthermore, *timescanner* does not actually extract MAC times from filesystems objects; for this, consider using The Sleuth Kit's *fls*, *ils* or *mac-robber* or TCT's *grave-robber* tool. Like *log2timeline*, *timescanner* reads various filesystem objects for data/time metadata contained therein, not metadata actually contained within the filesystem itself concerning said object. For this reason, The Sleuth Kit continues to remain an essential component of generating digital timelines and as such is an integral part of this work and the provided prototype (see Annex B.1). However, when the need for extracting date/time metadata from within certain files is necessary, then either *log2timeline* or *timescanner* should be used.

Ultimately, the choice of which tool to use, either *timescanner* or *log2timeline*, resides with the reader. The solution the reader chooses should best reflect his actual requirements. The authors' own specific requirements were based out of necessity in analysing several dozen disk images which required keeping a highly detailed track of each filesystem object and its associated date/time metadata. In the end, this required implementing a highly customized timeline format which could just that, and it has been implemented as prototype herein, based largely on The Sleuth Kit and *log2timeline*.

Moreover, in generating these timelines the authors came to the realization that little comprehensive information was available in the public space and what was accessible often left more questions than answers. As such, this work fills that void by providing useful information to the reader concerning the generating of customised digital timelines. In addition, the reader is provided with sufficient technical information that as an investigator he could readily explain the concepts behind timeline generation in a courtroom setting. Of course, discussing tool specifics will require examining that tool's source code prior to providing any such explanation which is not conducted herein.

1 Introduction

1.1 Objective

The objective of this technical memorandum is to introduce both relevant information concerning the use and generation of digital forensic timelines as well as the various challenges surrounding them. In addition, this technical memorandum will provide sufficient background material including example shell scripts and C code so that the user can successfully carry out and generate his own digital forensic timelines in a consistent manner. At the same time, the reader will be able to exploit additional sources of date/time-based metadata in order to increase his contextual understanding of transpired events as they relate to a given investigation.

1.2 Context

In the course of conducting a digital forensic investigation, many prospective sources of information are potentially available to the investigator. However, many sources of date/time-related metadata information are underutilized by investigators. These sources of information are found throughout suspect disk images and are too often overlooked for inclusion in a digital forensic timeline. Often times, however, timelines are not even used by investigators as the tendency in the digital forensics community is to forgo their use.

An important problem with digital timelines is that while they can appear very simple to assemble based on routinely available filesystem date/time-related metadata, much more data is often lurking beneath the standard filesystem[1]. However, in order to access a lower level of date/time-related metadata, it is necessary to search for both specific file types and locations which are known to contain the sought after metadata. Once extracted, the additional metadata may provide the investigator with a more reliable temporal context and frame of reference.

Although some automated digital forensic timeline-based generation software exists, as examined in Section 2.6, they all have certain caveats. To date, no automated super-timeline generation software is available. Thus, in order to improve digital timelines by incorporating additional date/time metadata, the investigator must know both the file types to include and their location in order to pass them on to one or more specialized parsers, which will return a timeline-based output.

However, the largest problem facing investigators, at least with respect to lower-level filesystem metadata, is finding additional sources of information and adequately parsing and processing them. A more in-depth analysis of the various challenges facing forensic investigators concerning timeline generation is addressed in Section 2.4.

1 The standard filesystem, as denoted herein, refers to the actual files which comprise said filesystem. However, it is not technically possible to go beyond the filesystem itself. Rather, each individual file is herein considered as a sub-filesystem node or leaf which itself may contain additional filesystem-specific date/time-related metadata which requires extraction and analysis using appropriate software tools prior to inclusion in a digital timeline.

1.3 Benefits of improved timeline generation

The successful use and inclusion of various sources of information relating to date/time-based metadata can greatly help the investigator to improve his contextual understanding of transpired events prior to the commencement of an investigation based on available digital evidence. Moreover, digital timelines which are based upon multiple information sources may help the investigator improve his correlational awareness of various actions undertaken by individuals or groups suspected in the alleged security incident.

Improved timelines based on additional information sources can help the investigator to differentiate between user actions and valid filesystem and operating system operations relating to the proper functioning and maintenance of the computer system. Valid system actions seen through the appropriate context will appear as visible background noise which can be filtered out using analytical data processing techniques, which although not examined herein, are highly beneficial to removing inconsequential filesystem related activities. Moreover, this supplementary information can allow the investigator to better target specific filesystem changes that more precisely fit the dates and times of the alleged security incident(s) to understand and contextualize them.

Through the use of multiple filesystem-based date/time-related information sources, the investigator will be able to build better digital timelines. This will enable the investigator to better correlate seemingly unrelated events which in fact are interconnected. Without additional information sources, these links would likely be imperceptible due to his lack of improved temporal and contextual awareness.

1.4 Notes

Although this technical memorandum specifically targets Windows systems, sufficient background information has been provided regarding UNIX and Linux so that the reader can better draw comparisons between these different systems and if need be, modify the scripts and source code provided herein to suit his own specific UNIX-based requirements.

2 Background information

2.1 About digital timelines

A digital timeline can be defined as the representation of filesystem time-based metadata described in a human-readable manner which contains useful information relating to a specific security event. It could also be defined as the chronological ordering of filesystem-related events as preserved by the filesystem's record-keeping constructs and metadata structures, which are extracted and presented to the investigator in a human-readable manner.

Although digital timelines are very useful sources of security-related information, they tend to be underutilized and underexploited for reasons explored throughout this section. While timelines are useful in bringing together various chronological events, it cannot be overstated that coaxing all the potential sources of information out of a filesystem is often difficult. Moreover, even once all the potential information has been queried and its metadata extracted and converted to a human-readable format, there is no guarantee that the filesystem's various time-based metadata sources have not been inadvertently altered or purposely tampered with.

While timeline generation software has been around for some time no intuitive GUI-based timeline visualization software yet exists due primarily to the difficulty in developing an application capable of responding to the needs of investigators when dealing with large datasets. The amount of data collected while generating a timeline can be large, upwards of several hundred thousands time-based events for an average modern operating system hard disk drive. When investigating multiple disks, it is easy to reach many millions of time-based events, although displaying them in an intuitive manner remains a challenging task.

However, the potential use for timelines is apparent. Filesystems and their time-based metadata can be a boon to the investigator by providing additional information with respect to changes within a given filesystem. Although timelines are a useful digital forensic construct, they all too often lack the inclusion of additional date/time-based information sources which can help the investigator better contextualize filesystem changes.

Important sources of date/time metadata are littered across a given hard disk drive. However, specific locations and key file types are likely to yield more tangible results than others. For example, consider the inclusion data sources such as the Windows Registry, Recycle Bin and various Windows log files, each replete with time and date information. Moreover, user interactions can be better understood by extracting date and times from sources including electronic mail, Internet usage based on cookies and cached files, chat software logs, etc. Coherently using as many date and time sources as possible may help the investigator build a comprehensive portrayal of events leading up to the security event in question.

2.2 Reasons for using open source software for timeline generation

2.2.1 Commercially used forensic analysis software background

Today's software market is littered with tools and programs which cater to the forensic investigator. Due to the plethora of available options choices need to be made. This is particularly important when considering how to best generate digital timelines from forensic disk images. Many potential disk image filesystems abound, although the most commonly analyzed are FAT and NTFS, typically from Windows-based computer systems. As such, it follows that the most commonly used software for analysing Windows-based forensic disk images are Guidance Software's EnCase and AccessData's FTK, both of which are particularly popular with law enforcement.

Although Guidance Software's EnCase is particularly well suited to developing additional forensic functionality through its built-in EnScript software programming language, EnScript programs can only run on computer systems where EnCase is installed. Whereas AccessData's FTK, which is another popular forensic software suite, it altogether lacks the software development features of EnCase. Its strong point, however, is its ability to simultaneously analyze multiple disk images via distributed computing[2]. Thus, while EnCase has the capacity to be readily expanded beyond its innate capabilities and FTK can scale to handle large investigations neither software product is adept at generating digital timelines.

Furthermore, both EnCase and FTK implement timeline generation and analysis differently. Unfortunately, the approach taken by each product is, in the opinion of the authors, altogether deficient and crude.

2.2.2 The use of Linux

The authors have therefore proposed that Linux should be used as a base platform for conducting timeline generation. This will, in the opinion of the authors, enable the forensic investigator to take advantage of the various data processing capabilities inherent to the operating system and take advantage of its unprecedented support for many disparate filesystems[3], unparalleled by any other operating system.

The use of Linux will enable the reader to develop a platform-independent and generic approach to timeline generation. In addition, any programs developed under Linux including Standard C Library-based [31, 54] C programs and those running under Bash[4], Perl[5] or Python will run with a minimum of effort and changes on other systems including Windows.

---

2 FTK version 3.2 is bundled with distributed processing capabilities.

3 It may be necessary to recompile the kernel in order to enable support for less commonly used filesystems.

4 Under Windows, Cygwin can be used to run Bash shell scripts.

5 The Cygwin environment for Windows can be used to run Perl scripts. However, Perl library dependencies may have to be satisfied, requiring the installation of additional library packages, as with any other computer system where Perl may be run.

While many different filesystems are in use today, those involved in the majority of forensic investigations are largely Windows-based filesystems including FAT12, FAT16, FAT32, NTFS and to a much lesser extent, exFAT. However, computer forensic investigations are not limited to these. Other lesser used filesystems occasionally acquired for forensic investigation can include: Ext2/3/4, UFS1/2, VxFS, FFS, RFS/ReiserFS, XFS, HFS+ and HFS, ZFS and Btrfs. These filesystems are used by a variety of operating systems including Linux, BSD, UNIX, Mac OS, Mac OS X and other UNIX-based systems.

Some filesystems are nearly universally[6] supported by modern operating systems and include FAT, CDFS (ISO 9660) and UDF filesystems. At the other end of the spectrum lie obscure filesystems supported only by proprietary hardware or electronic devices including handhelds (e.g. Blackberry, Palm, etc.) and other multimedia devices (e.g. TiVo, Xbox, etc.). Finally, there are virtual disk images[7] which can include Ghost disk images, VMware VMDK images, VirtualBox VDI images and Microsoft VHD images.

Since the majority of forensic investigations today are conducted against Windows-based systems it is important that any forensic software tool vying for market share be Windows compatible and support the various Windows filesystems used over the last two decades. While there was a time when commercial forensic software suites including FTK and EnCase supported only FAT and NTFS filesystems, today they each boast broad filesystem support even for non-Windows based filesystems[8] [9]. Even relative newcomer X-Ways Forensics boasts a broad list of supported filesystems[10]. Thus, support for commonly used filesystems is very important to timeline generation and analysis as this software is dependent on the filesystem support provided by either the underlying operating system or a specific software tool or suite.

Although various Windows-based commercial software offerings (e.g. EnCase, FTK, X-Ways, etc.) are highly capable in their own right, they are not suitable for the advanced data processing required for timeline generation and analysis in the manner the authors envisioned. However, Linux, with its advanced built-in[11] data processing capabilities and use of external third-party tools and scripts, including Brian Carrier's The Sleuth Kit, proves an ideal timeline generation platform.

---

6 This is applicable to most modern operating systems, Windows or UNIX-based or POSIX compliant [6, 7 and 14].

7 Software exists under Linux from QEMU and VirtualBox to manage and convert VMDK, VDI and VHD between each other. VMware Workstation-specific command line tools can be used to perform specific operations against VMDK disk images.

8 FTK 3.2 has support for FAT12/16/32, NTFS, Ext2/3/4, exFAT, VxFS, Microsoft VHD, AFF, EWF and Blackberry IDP backup files, not including the other disk image formats it supports. [2, 3]

9 EnCase 6.0 supports EWF, FAT12/16/32, NTFS, exFAT, Ext2/3, UFS, ReiserFS, JFS, FFS, Palm, HFS, HFS+, CDFS (ISO 9660), UDF (DVD) and TiVo 1 and 2 filesystems. [1, 3]

10 X-Ways Forensics supports FAT12/16/32, NTFS, exFAT, TFAT, Ext2/3/4, CDFS (ISO 9660), UDF (DVD) and Next3. [4]

11 These capabilities are inherent in not only Linux but in most modern UNIX-based operating systems.

2.2.3 Use of The Sleuth Kit

The fact that The Sleuth Kit is free and open source software by no means infers that it is inferior to its commercial counterparts (e.g. EnCase, FTK, X-Ways, etc.). In fact, for a free and open source software suite, it provides exceptional filesystem support[12] and forensic analysis capabilities and fully supports all the filesystems commonly required for investigating Windows systems. Moreover, The Sleuth Kit is an integral part of the timeline generation technique carried out in this technical memorandum.

The only notable exception to The Sleuth Kit's support of Windows-based filesystems is for exFAT, which is relatively new and not yet in wide use, although its acceptance is growing with the increased usage of large flash-based storage devices. However, the fact that some filesystems are not readily supported by The Sleuth Kit does not prevent it from being used under Linux. All that is required is that the kernel supports the filesystem, either as a natively compiled-in kernel module or as a FUSE module. Common FUSE-supported filesystems include exFAT and ZFS.

Unfortunately, the majority of The Sleuth Kit's tools will not work against disk images they cannot recognize. As such, disk images which are exFAT or ZFS-based, for example, cannot be directly analysed by The Sleuth Kit.

Although The Sleuth Kit fully supports CDFS, UDF is not supported. It is however important to consider that many optical discs, CD or DVD-based, are still based on the ISO 9660 standard instead of the more recent UDF standard[13] [6, 7]. For example, consider the various optical discs that the different versions of Windows (Windows '95 to Windows 2008) have been distributed on. They are all based on the ISO 9660[14] filesystem even though many of them have been written to DVD requiring more storage than afforded by typical CDs (see Annex A.1 for more details).

The ISO 9660 standard is still more readily used than UDF because it allows for filesystems much larger than the physical limitation of CDs (approximately 700 MB) with a maximum filesystem size of 8 TB [6, 52]. However, CDFS suffers from filename and directory size limitations. UDF, on the other hand, supports both larger file names and directory sizes [7, 9 and 14]. Moreover, a UDF filesystem can store an internal CDFS filesystem useful for creating hybrid filesystems which can coexist on the same optical disc [6, 7, 8 and 9]. As such, true DVD UDF-based filesystems seem to be employed less than common wisdom may suggest.

The situation is the same for non-Windows optical discs including those used for the installation of UNIX-based operating systems such as Solaris and Linux. While many Linux and BSD-based operating systems can fit on a single CD, many others are distributed by DVD due to their large size. Examples of this are available in Annex A.2.

---

12 Currently, as of version 3.2.0, The Sleuth Kit supports the following filesystems: Raw (dd-style disk images), AFF, EWF, NTFS, FAT12/16/32, UFS1/2, Ext2/3, HFS+, Swap and CDFS (ISO 9660) [5]. It does not yet support UDF or exFAT, both of which are supported by FTK, EnCase and X-Ways Forensics [1, 2, 3 and 4].

13 Using a DVD optical disc does not require the use of the UDF filesystem; CDFS is commonly used atop DVD optical discs.

14 As determined using the Linux commands *isovfy*, *mount*, *file* and *hexdump*.

While The Sleuth Kit includes many tools and utilities, only several of them actually pertain to timeline generation. Specifically, two filesystem-aware tools, *ils* and *fls*, fully support all the filesystems that The Sleuth Kit already does. However, when filesystems not directly supported by The Sleuth Kit are encountered, other tools can be used in lieu. If all that is needed from these tools is filesystem object date/time metadata extraction, then consider using *mac-robber* from Brian Carrier, the author of The Sleuth Kit.

Another similar tool is *grave-robber*, which is a part of The Coroner's Toolkit and written by Wietse Venema and Dan Farmer. The *mactime* tool present in both The Sleuth Kit and The Coroner's Toolkit is used to parse the output from either *mac-robber* or *grave-robber* and convert it into a human-readable text-based timeline.

All that is required for generating a timeline against a filesystem not supported by The Sleuth Kit using either *mac-robber* or *grave-robber* is some form of filesystem support which at a minimum must be read-only. Under Linux, filesystem support is provided through one of two mechanisms. The first is direct kernel support which implies that the Linux kernel has built-in support for a given filesystem and the other through the Linux FUSE subsystem (see Section 2.2.4). Simple techniques examined in [10, 11, 12 and 13] will help to ensure that the forensic examiner's efforts are profitable and productive when working with filesystems not supported by The Sleuth Kit.

2.2.4 Use of log2timeline

As with The Sleuth Kit, *log2timeline* is an open source software suite. Written by Kristin Gudjonsson, it is a software timeline generation suite. Through a set of built-in plug-ins for importing various data files such as Bodyfiles, network PCAP files, Windows event logs and many other types of log file formats it can export processed timeline to multiple formats including the Mactime, CEF, CSV and other timeline formats. For further import and export specifics refer to Section 5.2.1.

Log2timeline is unique in that in comparison to every other possible timeline generation tool currently in use today, it supports the largest gamut of import and export data file formats. However, it does not read in raw filesystem metadata and as such requires that this information has already been processed by software tools including *ils*, *fls*, *grave-robber* or *mac-robber*. However, when *log2timeline*'s import and export data files capabilities are combined with The Sleuth Kit's filesystem date/time metadata support, a powerful toolset becomes available to the digital forensics investigator.

For these reasons, it was decided early on that The Sleuth Kit and *log2timeline* would be combined to create a robust timeline analysis generation suite. Although not all investigations require filesystem timeline analysis, by nevertheless incorporating them into an analysis, improved contextual awareness can be gained by placing events, particularly log generated events, in a more structured framework.

Log2timeline technical specifics can be found in Section 2.6.4. However, since the suite does not fully support Windows registry timeline extraction, rather only specific portions of the registry (see Section 5.2.1 for more details) Harlan Carvey's *regtime.pl* Perl script is used to convert all registry entries with date/time-related information to a Bodyfile-based timeline.

2.2.5 Linux capabilities

The authors believe that Linux, with its multitude of built-in tools, utilities and other third-party software can provide a highly capable data processing and analysis platform. For example, various tools can be used to process both low and high-level data from suspect disk images while other software can be used to mount, copy, extract, verify and validate a disk image's contents. Other tools can then be used to sort, find, recover, undelete, search, hash, convert, analyse, extract date/time metadata and detect and compare a disk image's contents. And this can be done for many different image formats and filesystems. Consider that a relatively modern Linux operating system can readily support, natively [40], through its FUSE subsystem [39] or through commercial or open source software add-ons, the following physical filesystem formats: ADFS, AFFS, BeFS, Btrfs, CDFS (ISO 9660), Coherent, CramFS, EFS, exFAT, Ext2/3/4, FAT12/16/32, Files-11, HFS, HFS+, HPFS, JFS, MINIX, NTFS, QNX4FS, ReiserFS, Reiser4, ROMFS, SysV, UDF, UFS1/2, UMSDOS, VxFS[15], XENIX, XFS and ZFS. There are likely other filesystems which could be supported but may require additional effort to render functional under Linux.

Other Linux tools can be used to extract used, unallocated and slack disk space from suspect disk images. Moreover, advanced text processing, string extraction and pattern matching capabilities may be of interest to those processing large datasets. However, the list of potential forensic capabilities available from Linux is too lengthy to fully enumerate herein.

Of particular importance to the forensic investigator, especially for timeline generation and analysis is the ability to script multiple tools together. This enables the output or results of one tool to be fed into one or more tools to conduct additional work thereby implementing a rudimentary form of automation. Linux systems offer many possible shell scripting languages to choose from, some more optimized for intensive data processing and forensic-like work than others. In this technical memorandum, the Bash shell scripting language is used for simplicity and portability.

As with any other forensic task, repeatability is a must. As such, it is possible to have the Linux system record in a re-playable text-like file[16] all console-related I/O, whether typed in or displayed as results from a program. This will enable the investigator to ensure that any commands used and specified parameters are recorded saved for future use.

It is the authors' belief that the use of the command line, its tools and various parameters can improve efficiency, reliability and repeatability of the forensic investigative process, particularly for timeline generation, which is a highly automatable task. It is for these reasons that the authors have conducted their work atop the Linux operating system.

The commands, scripts and tools presented herein are bound together into a cohesive timeline generation system using rudimentary a UNIX shell script and several simple C programs.

---

15 Veritas software including VxFS filesystem support for Linux is now sold and marketed by Symantec Corp.

16 This is done using the Linux *script* command.

2.3 Why are digital timelines underutilized?

Digital timelines are unfortunately underutilized in digital forensic investigations. This is partly due to the poor support afforded to timelines by most existing software forensic frameworks[17]. Another problem is that many investigators do not have a good working knowledge of low-level filesystem mechanics and instead rely on graphical interfaces to automatically do the work for them, rather than performing key tasks manually.

Low-level filesystem forensics is a complex maze of metadata and filesystem structures which must be understood prior to analysing them. Although it is possible to drudge through simple filesystems such as FAT [19, 22 23 and 28], others such as NTFS [21, 25, 28 and 53] are too complex to merely labour through. And although there are software tools which claim to have low-level filesystem capability, few of them are truly able to analyse the structures and metadata therein.

The problem is further exacerbated by the fact that many filesystems are poorly documented, even commonly used ones. Furthermore, many filesystem specifications are written for engineers or software developers and not easily understood by those not well-versed in software programming or design. Many more filesystem specifications are simply unavailable to the general public including NTFS, currently the most widely investigated filesystem examined by investigators.

Fortunately, there are some excellent references to today's common filesystems including FAT [19, 22, 23 and 28], Ext2/3/4 [20, 24, 26, 27, 28, 29, 30, 35, 36 and 37], RFS [26, 27, 30, 32 and 33], UFS1/2 [26, 27, 28, 29, 30 and 38] and VxFS [26, 27, 29, 30, 35 and 41]. Some software, particularly The Sleuth Kit and AFF's *fiwalk* program, have exceptional low-level filesystem analysis capabilities for extracting metadata and other structure-related information. Other software suites such as FTK and EnCase do provide information concerning low-level filesystem metadata although not in the same manner. The metadata is presented as file-related attributes rather than treated as separate entities from the files themselves.

In contrast, through EnScript, EnCase has the ability to allow for the automatic collection of such metadata and structure-related information, information about such a script is not forthcoming at this time. FTK's filesystem metadata and structure analysis capabilities are no better than EnCase's.

Possibly the most important reason why timeline analysis has not caught on, is the inability for existing software to analyse, parse, visualize and present very large amounts of information to the investigator in a meaningful manner. While the timeline analysis of a simple desktop with one hard drive is manageable, the analysis of dozens of disks with multiple partitions with data spanning over many years can pose very significant data processing and analysis challenges. Processing and making sense of such information can take days and even weeks, especially if conducting pattern and trend analysis. However, the ability to visualize such vast amounts of data would be of significant help in piecing together the various events or actions sought after. Unfortunately, no known software tool exists in the realm of digital forensics which is capable of visualizing such vast amounts of data, correlating and finding important patterns and trends.

17 Consider FTK, EnCase, Autopsy and PTK to name a few.

Although certain data fusion products could be used, these products tend to be somewhat domain-specific. Thus, sometimes gargantuan efforts must be made to search for keywords, times and file names in order to attempt to make sense of the data.

However, when analysing very large disk sets, it is not uncommon for all the amassed timeline-related data to be many gigabytes in size and spanning millions of time-related events. Attempting to search through this data multiple times using standard data processing tools including *grep*, *cat*, *awk*, *sed* and *tr* can make analysis painfully slow. Therefore, better search capabilities are required. The use of databases, including *MySQL* and *PostgreSQL*, can significantly speed up searches. However, this requires that the investigator be knowledgeable in carrying out SQL queries and implementing the appropriate database structures and importing the data into said structures.

Thus, to make timeline analysis a more integral part of the investigation process, improved processes and tools are needed. This also requires enhanced text and data processing and visualization capabilities. In addition, pattern and trend analysis software would be highly advantageous when dealing with large volumes of data. However, these capabilities are not as of yet integrated into any forensic tool-set or suite known by the authors. In the future, it may be possible that software suites including EnCase[18] and FTK will at a minimum, provide a semi-automatic approach to conducting timeline generation and analysis with the ability to visualize the timeline.

2.4 Timeline challenges

The generation of accurate and pertinent digital forensic timelines can present many difficulties to the investigator. Many of these difficulties are technological in nature in that sufficient solutions are not yet available while others can be corrected with a change in perception by the investigator.

An important point is maintaining and preserving the chain of custody and chain of evidence. In order to do so, the investigator must never work directly with the evidence disk or even the first copy thereof (or file-based image copy of the evidence disk) but only with the second or tertiary copy of the original suspect disk. All disk operations must be read-only at all times. This can be achieved using write-blockers or using the Linux operating system *mount* command which can successfully maintain a designated read-only disk's integrity.

One very important challenge to consider is the fact that filesystem MAC time attributes can be directly modified by specialty software tools often employed by hackers to cover their tracks. These tools exist on the Internet and can be readily found in hacker forums; however, the knowledgeable hacker is likely to write his own software to evade detection from AV software and IDS/NIDS systems. Finding evidence of such tampering can be very difficult, especially if the software was operated remotely against a locally exploited machine.

Many modern operating systems, in their quest for improved filesystem performance, will allow the system administrator to disable access date/time updates for files and directories. Generally, operating systems will update each accessed file or directory's access date and time but if this

18 EnCase currently does support very basic timeline visualization capability but is extremely limited and provides little net benefit to the investigator.

feature is disabled, it will not be possible to know when a given file or directory was last accessed. It is a common practice under Linux and Windows to disable access update writes as it can significantly improve filesystem performance. Under Linux, disabling access updates must be manually configured on a per-filesystem basis. Under Windows, this became the default out of the box for Vista and Windows 7. Thus, not knowing when a file or directory was accessed can have an impact on the investigation when attempting to piece together from the timeline who accessed what. The advantage is that the investigator does not have to contend with matching file access changes to actions undertaken by AV scans.

While considering the ramifications of disabled access updates, it is important to consider the impact that some anti-virus scanners may have on a filesystem when generating digital timelines. Much modern AV software has been designed, to varying extents, to reset any changed access dates and times. The investigator may then have to contend with some scanners not resetting these appropriately and will have to remedy the problem through pattern and trend analysis to determine which access time changes were likely the results of an AV scan. Moreover, when considering that a digital timeline may contain hundreds of thousands or even millions of dates and times, any unnecessary additions caused by aberrant AV software are unwelcome.

Although this technical memorandum does not specifically explore the recovery of date/time metadata from deleted files, it is important for the investigator to know that this is possible and can provide additional sources of metadata information. The use of this additional metadata information can help investigators to better establish user patterns and trends based on historical metadata from user-based system usage. Despite the fact that some filesystems rely on inodes, other on inodes and extents and others on allocation tables, The Sleuth Kit can in many cases be used to read the date/time metadata of deleted files and directories which still have intact filesystem structures[19].

Other files that reside in unallocated space can sometimes be recovered using data recovery software with apparently intact date/time metadata. However, little credence should be placed on these recovered metadata since their integrity cannot be verified due to the absence of their original filesystem structures. If the investigator attempts to recover deleted files, he should concentrate on extracting useful date/time metadata from time metadata-rich files such as deleted event logs, registry hives, etc.

It is also possible, although not explored herein, to recover date/time metadata directly from recovered files that preserve their own date/time metadata within their own metadata structures (e.g. EXIF metadata within JPEG files). This is a time-consuming and complex endeavour likely to result in information overload. However, this approach may be important, especially when looking for specific file types. Thus, while the use of EXIF information is in interesting topic, it is not examined in this work.

Another important challenge rarely examined in digital forensics is the subject of NTFS Alternate Data Streams[20] (ADS) and their effect on filesystem date/time metadata. The trouble is that there is some uncertainty concerning what occurs to the file hiding the data stream when the stream is

19 This requires that inode or allocation table metadata and structures are still intact for a given file or directory.

20 NTFS ADS are sometimes used to hide data [42].

created and appended to the file as this subject is not found in the currently available literature. Thus, the authors undertook several short experiments to determine what changes, if any, are made when working with NTFS data streams.

Experiments conducted by the authors using NTFS-based files created at time X and then later appended with a data stream at time Y clearly show that file's modify and access date/time attributes have changed. Then, at some later time Z, simply reading the data stream was also found to update the file's access date/time attribute. However, the file's create date/time attribute remains unmodified throughout the actions undertaken at time Y and Z. Filesystem date/time attributes are specifically examined in Section 3.

Under Linux, using The Sleuth Kit and advanced data processing shell scripts, it is possible to detect and extract NTFS data streams. However, it is important to remember that copying or moving NTFS files with data streams to non-NTFS filesystems will not actually copy or move any appended data streams[21] (e.g. FAT and exFAT). Supplemental information concerning the effects of copying or moving NTFS files can be found in Annex A.3.

In today's modern computing world there are many sources of date/time information on a given hard disk drive. All too often, however, the vast majority of additional sources of date/time information are excluded from the overall timeline generation. This specific problem is limited to the commercial timeline generation tools including FTK, EnCase and X-Ways Forensics. The Sleuth Kit too does not have the ability to seek and incorporate additional date/time sources into its generated timeline. However, since the advent of *log2timeline*, another open source tool, it is possible to use The Sleuth Kit or The Coroner's Toolkit-based filesystem timelines and go far beyond their innate capabilities. *Log2timeline* date/time metadata source specific are examined in Section 5.2.1.

Another important issue which requires attention is the lack of a definitive digital timeline standard. Even though various tools can generate digital timelines, no official digital forensics timeline standard exists. The best standard to date, in the opinion of the authors, is that of the MAC time/Bodyfile format put forward by Wietse Venema and Dan Farmer and later improved upon by Brian Carrier[22] [10, 11, 13, 15, 17 and 18].

Finally, timeline generation can be a time-consuming process whose results are difficult to interpret. This is due to the amount of data processing required to extract date/time metadata from the filesystems and then seek out additional sources of information and coax any pertinent date/time information out of them. This information is difficult to understand due to the sheer number of timeline events which must be viewed in a linear and fluid manner. The use of improved visualization, pattern and trend analysis tools would be of great benefit here to the investigator in order to detect abnormal patterns and trends.

---

21 Some filesystems (e.g. Extent-based filesystems) when used in conjunction with an appropriate NTFS driver (e.g. *NTFS-3G*) can actually copy or move data streams into the filesystem space normally reserved for file-based extended attributes. However, this requires more testing which was not carried out by the authors in order to validate the maximum stream size which can be copied in to this space.

22 No official information can be found indicating that the standard actually begins with Venema/Farmer although no information exists to the contrary.

2.5 Brief remarks concerning the implementation of timeline generation

From the outset of an investigation, depending on the operating systems involved, it is important for the investigator to determine which data sources of date/time metadata are to be included in the final timeline. However, it is also important to choose which software tools or components will be used to generate the timeline. This too should also be based mainly on the operating systems and filesystems to be examined. Deciding upon these two key points early on will help the investigator formulate an appropriate plan of action for generating the timeline.

Regardless of which plan of action the investigator chooses, he must also consider if he will attempt to recover deleted data files which may be of use in the generation of a timeline. Operating system deletion of log files, registry hives, etc., are a normal part of operating system maintenance and as such recovering and including these files may help to add additional temporal context to the timeline. Recovering these files, however, requires an in-depth knowledge of data recovery tools and techniques which may not be limited to undeletion and data carving tools and software. [43, 44]

2.6 A short list and summary of available timeline generation software currently in use

The authors have compiled a representative list of digital forensic software tools and frameworks offering timeline generation capabilities in use today by forensic investigators. Moreover, the authors have provided a short timeline capability for each tool listed below so that the reader who perhaps is not familiar with some of the following tools better comprehend their timeline generation capabilities.

The most popular software timeline generation or visualization tools are, as determined by the authors in no order of precedence are as follows below:

- EnCase
- FTK
- Ex-Tip
- Log2timeline
- NTI FileList Pro
- The Sleuth Kit (*ils*, *fls*, *mactime)*
- Autopsy
- PTK
- Zeitline
- AnalyzeMFT.py
- Fiwalk
- Mac-robber
- DFF (Digital Forensic Framework)
- The Coroner's Toolkit (*grave-robber*, *mactime*)
- NFILabs Aftertime
- SIMILE Timeplot

A brief summary of the aforementioned timeline generation software tools follows below. All views and opinions expressed herein are those of the authors.

2.6.1 Encase

EnCase, developed and marketed by Guidance Software, is the world's premier commercial forensic software suite and also its most commercially successful one. It is the frontrunner for submitting processed digital evidence in court. It is difficult to determine how widely used it is but given EnCase's wide reaching acceptance in both the digital forensic community and the marketplace, it is likely that many forensic investigators are comfortable with this particular framework. The framework is filesystem-aware but does not read or write Bodyfiles.

2.6.2 FTK

FTK, developed and marketed by AccessData, is another favourite commercial forensic software framework popular with both law enforcement and government. It has a basic timeline generation capability that is no better than EnCase's. The framework is filesystem-aware but does not read or write Bodyfiles.

2.6.3 Ex-Tip

Ex-Tip, developed by Michael Cloppert, is a proof of concept tool submitted to SANS and the precursor, at least technologically, to *log2timeline*. While it does not read a filesystem for date/time related metadata, it can read in Bodyfiles and extract date/time metadata from McAfee log files and Windows registry. It has the ability to read and write Bodyfiles. The tool does not consider the underlying filesystem type as it must be mounted in order for this tool to work. [67]

2.6.4 Log2timeline

Log2timeline is the premier timeline generation framework. It supports the largest array of date/time metadata extraction capabilities (see Section 5.2.1 for a complete list) of all the programs and tools examined in this section. Moreover, it is a direct competitor to NFILabs Aftertime. The tools comprising *log2timeline* are written in Perl and will run on Linux, Mac OS X and other UNIX systems. It works by applying selected plug-in modules against specified files. Some of the tools and plug-ins will run on Windows while others require modifications to do so. The *log2timeline* framework is easily scripted to conduct advanced file detection and date/time metadata extraction. Although the tool comes equipped with its own recursive file processing tool, *timescanner*, which can parse every encountered file with selected plug-ins, this proves too blunt of a search. File searching and metadata extraction can become refined through scripting. This tool has the ability to read Bodyfiles. *Log2timeline* does not consider the underlying filesystem type as it must be mounted in order for this tool to work. [68, 69]

2.6.5 NTI FileList Pro

NTI FileList Pro is a commercial software file inventorying software system which is compatible with older versions of Windows, XP and previous versions including NT, 98, 95 and DOS. This

program does not currently appear to be in high use by government or law enforcement. This program is DOS-based and partially filesystem-aware. It does not read or write Bodyfiles.

2.6.6 The Sleuth Kit

The Sleuth Kit is the most widely used open source software forensic framework whose capabilities are supported both by PTK, *fiwalk* and Autopsy (to name only a few). It is very popular even among law enforcement officials, who use it to validate their own work carried out under EnCase and FTK. The Sleuth Kit relies on *fls* and *ils* to extract date/time metadata from filesystem objects and can write their output to the Bodyfile timeline format. On the other hand, the *mactime* tool is used to convert Bodyfiles into human-readable text. As a note, The Sleuth Kit's *mactime* tool cannot read metadata directly from a filesystem; instead, it must have a Bodyfile to read. The Sleuth Kit is filesystem-aware. [70, 71, 72 and 73]

2.6.7 Autopsy

Autopsy, the default graphical interface for The Sleuth Kit, provides the investigator with the ability to generate timelines based on the underlying tools found in The Sleuth Kit. Autopsy, which relies on The Sleuth Kit, is filesystem-aware and has the same Bodyfile and Mactime timeline format capabilities as The Sleuth Kit. [70, 71, 72 and 73]

2.6.8 PTK

PTK, developed and marketed by DFLabs, is the premier graphically-based commercial open source software offering. Its functionality is almost entirely derived from The Sleuth Kit although other functionality is made available by other open source forensic software tools and programs. Its timeline-based analysis capabilities are superior to those thus far enumerated herein as its timeline interface is easy to visualize and is somewhat intuitive to navigate. PTK, which relies on The Sleuth Kit, is filesystem-aware. It relies on The Sleuth Kit and Bodyfile format to generate its visual timelines.

2.6.9 Zeitline

Zeitline is a prototype digital timeline framework. It is written entirely in Java and should therefore be cross-platform compatible. It can import syslog-like files and date/time Bodyfiles generated using The Sleuth Kit's *fls* and *ils* programs (or from other Bodyfile-generating tools and programs). Zeitline can be used to process, filter and move various events around. It has no visualization capability. However, this program is not filesystem-aware.

2.6.10 AnalyzeMFT.py

AnalyzeMFT.py is a unique tool in that it is designed specifically for analyzing NTFS filesystems. What sets it apart is its ability to verify whether a filesystem object's creation time is intact based upon its examination of a lesser used NTFS creation time metadata structure. It can write its output to the Bodyfile format. This tool is NTFS-aware.

2.6.11 Fiwalk

Fiwalk, short for file inode walker, is a program which uses The Sleuth Kit to process a given disk image. It has the same filesystem support as The Sleuth Kit and can output an extensive set of data about all files and streams attached to a given filesystem. It can output all collected information to the console, to an XML file or generate a Bodyfile which can later be processed with *mactime* to generate a human-readable text-based timeline. *Fiwalk*, which relies on The Sleuth Kit, is filesystem-aware. [77]

2.6.12 Mac-robber

Mac-robber, written by Brian Carrier of The Sleuth Kit, is designed after The Coroner's Toolkit's grave-robber tool. The *mac-robber* tools reads in using standard system calls (under UNIX the *stat()* system call) the date/time and filesystem permission metadata concerning one or more filesystem objects. Output is sent in Bodyfile format to the console or redirected to a file for storage. *Mac-robber* does not consider the underlying filesystem type as it must be mounted in order for this tool to work. [13, 12, 74 and 75]

2.6.13 Digital Forensic Framework

DFF (Digital Forensic Framework) is an open source analysis framework which supports timeline generation and visualization. Both Linux and Windows precompiled binaries are available for use as is the source code for recompilation. The program supports FAT12/16/32, NTFS and Ext2/3/4. Its timeline generation is basic and supports only filesystem object MAC times. Its visualization capabilities are navigable but not as advanced as those provided by NFILabs Aftertime. This tool does not have the ability to save its work, print reports or generate Bodyfile Mactime or TLN timelines. Moreover, the timeline navigation window's export feature does not currently work. This program is filesystem-aware.

2.6.14 Grave-robber

Grave-robber (which actually calls The Coroner's Toolkit *mactime* program) is very similar to Brian Carrier's *mac-robber* tool, with the same basic features and functionality. This program is used far less often today as it has been altogether superseded by The Sleuth Kit and *mac-robber* and as such its use is not advised. *Grave-robber* and *mactime* do not consider the underlying filesystem type as it must be mounted in order for these tools to work. It can also save its output using the Bodyfile format. [13, 11, 10 and 76]

2.6.15 NFILabs Aftertime

NFILabs Aftertime, developed by the Netherlands Forensic Institute, is the most mature and capable timeline generation and visualization program available today. It is similar in capability to *log2timeline* but does not require the use of scripts to find and feed specific file types for date/time metadata extraction. Although it currently supports many file formats it is second only to *log2timeline* with respect to its number of supported formats. The tool is freely available to anyone and executable binaries are available for both Windows and Linux; however, no source

code is available for recompilation. Its visualization capabilities are superior to all other tools compared herein. The program only accepts disk images for timeline analysis and supports FAT16/32, NTFS, Ext2 and HFS. It supports both raw (*dd*-style) and EnCase-based disk images. This program is filesystem-aware. The tool's downside is that it cannot export its timeline to the Bodyfile, Mactime or TLN timeline formats and its HTML and CSV report generation functions currently generate only empty files.

2.6.16 SIMILE Timeplot

The final examined in this work is the SIMILE *Timeplot* widget which was originally written by Stefano Mazzocchi. It is a unique tool but requires much work to render it functional. It will work with simple timelines which have been stripped of all comments and unnecessary information, but this requires a script of custom tool to reparse a large timeline file into a more appropriate format which the widget can handle. Underneath the hood, it runs Java which makes enables it to be multi-platform. However, since Java can be very memory intensive, it can take a very long time before anything is rendered when working with many tens of thousands of points. Upon parsing the data into a usable format and then attempting to work with millions of points, the application crashed. Further adding to the frustration of using this tool is the necessity of customizing the widget to specific data formats and adjusting the output. This widget only supports its only timeline format which can otherwise be generated by *log2timeline*.

3 Filesystem-specific details

3.1 Background

In this section, various technical details are examined concerning the various filesystems investigators are most likely to encounter. The discussion will include an in-depth treatment of MAC times and filesystem permissions as they pertain to FAT, NTFS and UNIX-based filesystems.

3.2 Topics excluded from the filesystem analysis

3.2.1 Specific filesystems

Various filesystems have been left out from examination in this section. For Windows-based systems, CDFS, UDF and exFAT have been excluded. For UNIX and Linux, the default filesystem Ext2/3/4 has been assumed as the underlying filesystem the investigator will analyze. The reasons for these exclusions are examined herein.

Where optically-based filesystems are concerned (CDFS and UDF), once data is written to a CD or DVD, all data written to the disk remains intact unless the disk is reformatted using CD-RW, DVD-RW or BD-RW technology. Although optical discs can be appended to and can even have new volumes added to them which can be set to disregard all previous volumes on a given optical disc, the fact remains that other volumes and their data continue to persist even if they are not immediately accessible. Special software exists in both the commercial and open source world which can read any and all optical disc volumes even if those volumes are no longer recognized by the underlying operating system or optical disc device driver. That said, once those volumes are read, their date/time metadata can be extracted using software such as *mac-robber*. [6, 7, 8, 9, 52, 57, 58 and 59]

As for exFAT, a recent filesystem from Microsoft intended specifically for flash storage, it can nevertheless be used on mechanical disk drives. However, unlike the FAT filesystem which has a publicly available specification [19, 23 and 28] and NTFS, which has been reverse-engineered [21, 25, 28, 53, 63 and 65], no publicly available Microsoft specification exists for exFAT in order to shed light on its underlying metadata structures. Fortunately, a recent open source software initiative by Andrew Nayenko (*FUSE exFAT* and *exFAT Utils*) provides functional source code which supplies both a FUSE module for mounting exFAT volumes and various utilities for managing said volumes. In order to understand the underlying capabilities and features of this new filesystem, the investigator will have to study the source code. Fortunately, however, software date/time metadata extraction tools such as *mac-robber* can be used against exFAT. Although exFAT is gaining in popularity, it is not yet popular enough, at least in the opinion of the authors, to warrant direct examination in this technical memorandum. [60, 61]

For the UNIX-specific analysis of this section, the information has been presented in a filesystem-neutral language. Although the Extended filesystem is the default Linux filesystem in most

distributions, Linux is capable of working with many diverse filesystems from a range of operating systems.

3.2.2 Extended attributes

Many modern filesystems support additional filesystem metadata structures that include extended attributes. Windows supports extended attributes only for NTFS which can be read and set directly from Linux. While modern UNIX operating systems including Linux fully support extended attributes working with them under UNIX and Linux, it is more complicated than working with NTFS extended attributes. As such, Linux and UNIX and extended attributes are not examined in this work.

The problem in working with them under Linux is that the issue lies in the fact that many other filesystems' extended attributes are supported using proprietary software tools and UNIX kernels. For example, although modern Solaris implementations fully support extended attributes, accessing them under Linux is not possible because it has neither the appropriate software tools nor does it have the necessary UFS kernel filesystem structures to support Solaris-based extended attributes, as determined by the authors through experimentation. However, other filesystems including Ext2/3/4, XFS, JFS and Btrfs fully support the reading and setting of extended attributes under Linux, as also determined by the authors through experimentation.

3.2.3 A note about ACLs

Both Windows and Linux support filesystem ACLs. As such, a short examination is appropriate. Both systems provide the necessary software tools required to work with ACLs. Moreover, Linux can not only work with Linux and UNIX filesystem ACLs but can also work with NTFS ACLs. Although ACLs are not currently used for building digital timelines their inclusion has the potential to help the investigator better understand the links between filesystem resources and those who have access to them. Conversely, the use of ACLs in such an analysis may introduce new sources of information which can lead to confusion and information overload. The inclusion of ACLs in digital timeline analysis has not been carried out herein.

3.3 About MAC times, their limitations and consequences

3.3.1 Background

MAC times, sometimes also called MACB times, are the date/time filesystem metadata associations used to express the specific date/time metadata different filesystems use. These expressions are used by both the Bodyfile and TLN formats and are represented as *m*, *a*, *c* and *b*, respectively. More specifically, *M* corresponds to a filesystem unit's[23] modification time and is often denoted as *m* and referred to as *mtime*. A filesystem unit's last recorded access time, *A*, as represented by *a* is commonly referred to as *atime*. The last change or modification date/time

23 A unit is a file, link, directory, pipe, device or any other type of valid filesystem object.

recorded for a given filesystem unit, *C*, is represented as *c* and referred to as *ctime*. Finally, a filesystem unit's creation time, *B*[24], commonly referred to as *crtime*, is denoted as *b*.

However, different filesystems and operating systems behave differently and may not necessarily implement the aforementioned range of MAC times in the same manner. However, other than *crtime* for NTFS, no notable examples have been found. [10, 11, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 28, 29, 32, 33, 41 and 53]

3.3.2 MAC times under Linux and UNIX

As far as it can be determined, commonly used Linux and UNIX filesystems, including Ext2/3/4, UFS1/2, VxFS, RFS, etc., implement the *mtime*, *atime* and *ctime* date/time metadata constructs [20, 24, 28, 29, 30, 31, 32, 33 and 41]. In fact, because modern UNIX and Linux systems are based on the ANSI Standard C Library [31, 54] and other POSIX [55] compliant C libraries, there is uniformity across these operating systems. Filesystem uniformity is achieved using the UNIX *stat()* system call which is used across Solaris, Linux, BSD and others [29, 30 and 31]. Most, if not all modern UNIX-based filesystems, will behave similarly with respect to MAC times. Precise details are not available for some filesystems as their specifications (e.g. XFS, JFS, ZFS, etc.) are not available in the public domain. While their source code is in the public domain, a complete source code review is necessary in order to understand the internal mechanisms of these filesystems and their similarity to those mentioned above.

UNIX and Linux supported filesystems with publicly available design specifications include FAT[25] [23], Ext2/3/4 [24] and RFS [33] while other filesystems including UFS1/2 [29] and VxFS [29, 41] are sufficiently well documented that an investigator can find his way around and understand the underlying functionality.

The Linux Ext2/3/4 filesystems are similar to their brethren (UFS, VxFS and RFS) except for the fact that the former provides a unique date/time metadata attribute known as *dtime*[26]or deletion time [20, 24, 28 and 29]. No other known UNIX filesystem supports this specific attribute. Unfortunately, none of the tools listed in Section 2.6 actually supports the *dtime* attribute, thereby requiring that this metadata be extracted using custom software or manually with a hex editor. Irrespective of the *dtime* attribute, these filesystems support the *mtime*, *atime* and *ctime* attributes due to their reliance on the UNIX *stat()* system [29, 30].

According to C library file *sys/stat.h* and its dependencies, *ctime* denotes a change in a file's status, which occurs whenever a file's contents are changed or modified. However, *ctime* modifications also occur when there is a change to a filesystem object's metadata status, including changes to file permissions, file ownership, etc. [30]

On the other hand, the *mtime* attribute is updated whenever there is a change to a file's contents and any update to *mtime* almost always ensures an update to *ctime*. While this may seem

---

24 *Crtime*, or creation time, is native only to the NTFS filesystem [17, 21, 25, 28 and 53].

25 Although FAT is a Microsoft specific filesystem, most modern UNIX-like systems support it natively.

26 Under Fedora Core 14 64-bit, *dtime* is defined by C library files *linux/ext2_fs.h* and *linux/ext3_fs.h*.

confusing, it is important to think about it in terms of data and metadata. *Ctime* reflects metadata changes while *mtime* reflects changes in data. A change made to a filesystem object's metadata does not generally change the data physically residing in the filesystem object; however, changes made to physical data will usually result in changes to the metadata[27]. More information about UNIX-based file deletion can be found in Annex A.3, tables 7 and 8. [20, 24 28, 29, 30, 32, 33 and 34]

In order to improve filesystem performance, some UNIX operating systems, including Solaris and Linux, allow the system administrator to disable filesystem access time updates (*atime*) on both directories and files [29, 30 and 40]. Under Linux, this is accomplished on many supported filesystems including FAT, NTFS, Ext2/3/4, XFS, JFS, RFS, UFS1/2, etc., using the *nodiratime*[28] and *noatime*[29] mount options [29, 40]. Other UNIX operating systems may use the same or similar mount options.

UNIX file date/time and metadata is stored in epoch time and as such is not time-zone dependent. The date/time metadata is presented to the investigator in UTC and converted by various date/time handling software into local time. Thus, whatever actual time a given file has, it will always be treated uniformly. [34]

3.3.3 MAC times under Windows

3.3.3.1 FAT-based MAC times

The FAT filesystem has been around for many years and is a well-documented filesystem [19, 22, 23 and 28]. It is similar to UNIX filesystems with respect to its date/time metadata attributes for both files and directories. It too only uses *mtime*, *atime* and *ctime*. However, FAT's *atime* metadata attribute can only be expressed by date but without any time-specific information[30] [17, 19, 22, 23 and 28].

It is important to note that the FAT filesystem is not a highly time-accurate filesystem. Specifically, *mtime* or *ctime* metadata written to the filesystem are likely to be inaccurate because FAT-based *mtime* and *ctime* entries may be off in time by upwards of one or more minutes. It is also important to note that FAT time is locally based and therefore does not preserve any UTC context. As such it is entirely dependent on the suspect system's geographical location. [19, 22, 23 and 28]

3.3.3.2 NTFS-based MAC times

The most commonly examined filesystem today is NTFS, as it is the default filesystem for all versions of Windows since Windows NT [21, 25, 28 and 53]. However, it is also the most complex filesystem in use today due to its excessive internal data structures [21, 25, 28, 45, 46

27 E.g. The file is now larger, smaller, different permissions, etc.
28 *nodiratime* denotes do not update directory access times.
29 *noatime* denotes do not update file access times.
30 As such, *atime* is limited to year, month, day but no time (Hours:Minutes:Seconds) is encoded for this attribute.

and 53]. The NTFS file metadata structure is very different from other filesystems, especially UNIX-based filesystems, which rely on inodes or extents [20, 24, 26, 27, 28, 29, 32, 33, 35, 37, 38 and 41]. NTFS, on the other hand, relies exclusively on the MFT [21, 25, 28, 45, 46 and 53] and without it the NTFS filesystem is rendered useless. Nonetheless, the NTFS filesystem does make use of extents [53].

Under NTFS, every file and directory has an MFT entry, however, ADS streams rely on the MFT entry of their host files. Moreover, each file and directory has two different MFT date/time-related entries, specifically the $FILE_NAME and $STANDARD_INFORMATION attributes, which each contains the file's or directory's *mtime*, *atime*, *ctime* and *crtime* metadata.

The $STANDARD_INFORMATION construct is the primary date/time metadata structure used both by the operating system and the vast majority of applications. When a file or directory's date/time information is queried through the command line using the *dir* command or through *Windows Explorer*, it is the date/time information recorded within $STANDARD_INFORMATION which is presented to the user. While The Sleuth Kit and some of the other tools presented in Section 2.6 rely specifically on this date/time structure, others including *analyzeMFT.py*[31] and *ntfsinfo*[32] can read both. However, timeline analysis is conducted primarily using $STANDARD_INFORMATION. [21, 25, 28, 42 and 53]

There is one very good reason for using both $STANDARD_INFORMATION and $FILE_NAME date/time metadata in timeline generation. While the former is the primary date/time metadata, the latter provides a point of reference against which to verify a given file's creation date. This is particularly important to consider if there is evidence or suspicion of aberrant or malicious filesystem usage as the corroboration between an NTFS object's $STANDARD_INFORMATION and its $FILE_NAME date/time metadata may be able to determine the file's true creation date. [21, 25, 28 and 53]

It is common today for malicious software and remote attackers to cover their tracks by changing file dates and times including creation time. These date/time metadata changes are almost always made to $STANDARD_INFORMATION and not to $FILE_NAME. Thus, when a given file's date/time is suspect, its "backup" date/time metadata ($FILE_NAME) can be used as a more reliable source of the file's actual creation time. To carry out this analysis, consider the aforementioned tools *analyzeMFT.py* and *ntfsinfo*. [21, 25, 28 and 53]

However, $FILE_NAME-based *crtime* date/time metadata can sometimes be modified when basic file operations are performed and as such the investigator must understand this clearly. For example, when an NTFS-based file is moved to an altogether different NTFS volume, its $FILE_NAME-based *crtime* is updated to reflect the current system time. On the other hand, if the file is copied either to the same or to an altogether new volume, then its $FILE_NAME-based *crtime* is again updated to reflect the current system time. However, if the file is only moved elsewhere on the same NTFS volume then its $FILE_NAME-based *crtime* remains untouched. [21, 25, 28 and 53]

---

31 The program provides MFT *crtime*-based anomaly detection as of version 1.7.

32 Although this program does not provide bodyfile or TLN output, it is possible through shell scripting to implement anomaly detection.

Furthermore, while the $FILE_NAME MFT structure is useful for *crtime*-related changes, it is not useful for verifying a file's actual *mtime*, *atime* or *ctime* as $FILE_NAME's date/time metadata is rarely updated. As such, it will likely not reflect date/time metadata changes as they pertain to file access, modification and change attributes. [21, 25, 28 and 53]

Unfortunately, due largely to its lack of publicly available documentation, the complexity of the MFT and other NTFS filesystem structures are not well understood; even the best sources of information fall short [21, 25, 28 and 53]. Some have attempted to study the effects of filesystem modification and determine the impact on the date/time metadata. For example, consider the work of Rob Lee[33] from the SANS Institute, who attempted to thoroughly understand and work through the date/time metadata impact of filesystem changes and which can be found in Annex A.3, tables 5 and 6 [16]. Then compare said studies against those of Brian Carrier, which can be found in Annex A.3, tables 7 and 8 [28]. Upon examining the aforementioned tables, confusion ensues, thus leaving the reader and the authors wondering who was right, Lee or Carrier. For this reason, additional research must be made into NTFS filesystem modifications and their impending date/time metadata consequences.

Further exacerbating the problem with NTFS is that both it and its operating system (Windows) have continued to evolve but with no documentation specifying what exactly has changed in NTFS. It is certain that the version of NTFS now in use under the latest incarnations of Windows is not the same as the original version, which first came about in the early 1990s. Without any publicly available specification, it is not possible to definitively determine which changes have occurred without reverse engineering the filesystem. Moreover, Windows conceals its underlying functionality from the programmer through the use of ambiguous APIs and ABIs.

3.3.4 Final thoughts

Most of the filesystems examined in this section have an easy to comprehend date/time metadata structure. For some which have publicly available specifications, including FAT, Ext2/3/4 and RFS, they are readily understood. Other UNIX-based filesystems which lack publicly available specifications instead provide source code which can be studied in order to piece together their functionality and capability, which includes filesystems such as XFS, JFS and ZFS. Specific academic references exist which provide the investigator with sufficient details concerning filesystem internals that he will undoubtedly comprehend its functioning, which includes filesystems such as UFS1/2 and VxFS. In short, these filesystems can be sufficiently well understood by an investigator that he can, upon rigorous study, explain them to others, including a courtroom.

On the other hand, while NTFS is a modern and capable filesystem, it is as far as could be from FAT or any UNIX filesystem. It has many similarities to OpenVMS's native filesystem, Files-11 [45, 46]. Reasons exist for these similarities but are outside the scope of this work. Nevertheless, the excessive complexity of the NTFS filesystem obfuscates features and functionality from the investigator which he can only begin to comprehend upon studying the source code of various open source NTFS-based initiatives including the *NTFS-3G* driver, *ntfsprogs* and programming guides such as [25].

33 His work presented in [16] has been presented by an altogether different individual although it is entirely based on his own work.

In the opinion of the authors, the underlying features of many filesystems including FAT, Ext2/3/4, RFS, UFS1/2, VxFS, XFS and JFS can be explained with confidence in a courtroom. However, without examining the aforementioned sources of information concerning NTFS, investigator will likely not be able to successfully explain its capabilities and intrinsic features in court to a literate jury.

3.4 About file permissions

3.4.1 Background

Different filesystems rely on differing methods for securing files and data. Some rely on simple attributes, many fully support permissions (commonly known as DACLs), others access control lists (ACLs) and finally, some support extended attributes. Of course, it is entirely possible for a filesystem to support many or all of the aforementioned permission attribution mechanisms.

It is not generally possible to accurately represent permissions from non-UNIX filesystem objects while generating a timeline using software originally designed to function under UNIX including The Coroner's Toolkit (e.g. *grave-robber*, *mactime*), The Sleuth Kit (*fls*, *ils*, *mactime*) and AFF *Fiwalk*. However, none of the Windows-capable tools examined in Section 2.6 fares any better at representing Windows-based filesystem permissions than their UNIX counterparts.

3.4.2 A timeline-based representation of permissions

Although different filesystem permission mechanisms may be at play on a given filesystem, not all permissions are necessarily mapped into corresponding timeline permissions. It is important to understand that the specific tools used herein[34] to address the generation of enhanced timelines were not specifically written for Windows or DOS. Instead, they were originally designed for UNIX-based systems and as such, any timeline generated using the techniques and information presented herein will use the UNIX-based permission system. UNIX permissions[35] are based on *read* (*r*), *write* (*w*) and *execute* (*x*) for users, groups and the public (e.g. *rwx-rw-r-x*).

Some permissions are readily mappable to UNIX-based authorizations but most are not. For example, when considering DOS attributes (see Section 3.4.3 for more details), enabling the read-only attribute will cause the timeline generation software to set the displayed permissions to read-only, while all the other attributes will have no effect. However, since DOS files do not record users or groups (UIDs and GIDs), the timeline generation software will display read-only for both the file's group and public permissions. A helpful clarification is in order. Consider the following examples concerning the various possible DOS attributes for several files:

Ex. 1)	*FileA.txt* attributes	+R +H +S +A
	Interpreted timeline permissions:	-r--r--r--
Ex. 2)	*FileB.txt* attributes:	-R +H -S +A

34 The tools used herein are also the most capable as compared to the others presented in Section 2.6.
35 Additional permissions including SUID/SGID and world-writable sticky-bits are not examined.

Interpreted timeline permissions: -rw-r--r--

Ex. 3) *FileC.txt* attributes: -R +H +S -A

Interpreted timeline permissions: -rw-r--r--

While DOS attributes are easy to translate to UNIX-based permissions, the additional attributes provided by NTFS may pose significant translation difficulties. Of course, UNIX permissions are correctly read and interpreted by the aforementioned software tools (see Section 3.4.1 for more details). However, ACLs and extended attributes both from NTFS and various UNIX filesystems cannot be adequately represented by the simplistic permission representation system used by UNIX. Since The Sleuth Kit and *log2timeline* are the cornerstones of timeline generation, for this technical memorandum all permissions will be represented using UNIX-based permissions. Other non-UNIX software may have different capabilities for representing filesystem object permissions but they are not examined herein.

3.4.3 Windows filesystem permissions

3.4.3.1 FAT filesystem permissions

The FAT filesystem relies exclusively on file attributes. Specifically, the standard DOS file attributes are the *Hidden* (*H*), *System* (*S*), *Archive* (*A*) and *Read-Only* (*R*) attributes. However, FAT does not allow for UNIX-style file permissions which set authorizations for groups, users and the public. Thus, FAT implements a rudimentary permission mechanism. Under DOS, permissions can be changed using *attrib* command[36]. [19, 23, 47 and 48]

Although FAT and DOS do not support any filesystem structure other than files and directories, other non-DOS operating systems may use modified versions of FAT which allow for a wider assortment of filesystem structures. Consider that while FAT does not support extended attributes, this does not stop them from being used by operating systems including OS/2, which does support FAT-based extended attributes which are stored in a specifically designated file residing on the FAT-based filesystem in question. However, it is important to note that the appropriate application of OS/2 FAT-based extended attributes is available only under OS/2 running its proprietary kernel and FAT filesystem driver. [19, 62 and 64]

3.4.3.2 NTFS filesystem permissions

NTFS filesystems support DOS attributes, extended attributes and ACLs [21, 25, 53 and 63]. Windows, like UNIX, is a DACL-based operating system where DACLs are permission-based and are applied to files and directories (and other system objects) by the user or administrator [21, 25, 47, 50, 53 and 63]. There exist five basic file permissions which are used to configure a given ACL which are as follows [65]:

- Write
- Read

36 All versions of Windows support this command from a DOS prompt command line window.

- Read & Execute
- Modify
- Full Control

However, a more fine-tuned set of file permissions are available if required and are as follows [65]:

- Traverse Folder/Execute File
- List Folder/Read Data
- Read Attributes
- Read Extended Attributes
- Create Files/Write Data
- Create Folders/Append Data
- Write Attributes
- Write Extended Attributes
- Delete Subfolders and Files
- Delete
- Read Permissions
- Change Permissions
- Take Ownership
- Synchronize

Additional permissions are available for non-filesystem objects including the registry, printers, system services and network shares. All filesystem ACLs can be allocated by users, groups and other objects such as computers in a domain or Active Directory Organizational Unit. NTFS also provides ACL inheritance, a feature which enables file and directories to inherit the ACLs of higher-level directories[37]. This feature can also be blocked on specific files and directories. [65]

The NTFS filesystem supports several additional filesystem attributes more than DOS. It supplies the standard *Hidden* (*H*), *System*[38] (*S*), *Archive* (*A*) and *Read-Only* (*R*) attributes but also provides at least three other attributes including *Indexing* (*I*), *Compressed* (*C*) and *Encrypted* (*E*). There may exist additional attributes that are hidden although this can not be confirmed since the NTFS specification is not public. Two other attributes not visible to the user or administrator under normal circumstances, even from most software tools, are the *Temporary* (*T*) and *Offline* (*O*) attributes [63].

From the command line, most of the filesystem attributes including *H*, *S*, *A*, *R* and *I* can be changed using the *attrib* program. The others, *C* and *E*, can be changed through *Windows Explorer* where one or more files or directories can be selectively compressed or encrypted. Attributes *T* and *I* cannot be changed except by very specific software. Interestingly, however, using the Linux extended attribute modification tools, *getfattr* and *setfattr*, it is possible to not only manipulate all NTFS filesystem attributes but also change all of them [63]. Linux tool *ntfsinfo* has the ability to read all NTFS filesystem objects' attributes but it cannot change them. The aforementioned tool also does not have ability to read ACLs or permissions.

37 Inheritance can be blocked or set to an altogether different set of ACLs.
38 The presence of this attribute indicates that the filesystem object in question is a link [63].

ACLs can be set in Windows using command line tools including *cacls, icacls*, *SubInACL* and *SetACL* [47, 48, 49, 50, 51 and 53]. And of course, all attributes (except *T* and *I*) and ACLs can be set via *Windows Explorer*.

Although NTFS stores extended attributes, it does not use it for day to day operations. Instead, they were used in previous versions when Windows maintained compatibility with OS/2, which it no longer does. However, the extended attribute structures appear to have remained although what purpose they serve today is not entirely certain. [21, 53, 62 and 63]

3.4.4 UNIX filesystem permissions

UNIX and Linux file permissions are simple in comparison to NTFS' approach to permissions as examined in the previous subsection. Universally implemented by all modern UNIX-like systems are three basic permissions: *Read* (*r*), *Write* (*w*) and *Execute* (*x*). These permissions are applicable to almost all filesystem objects including files, directories, pipes, sockets, devices, etc. The only notable exceptions concern file and directory links and certain kernel objects which are generally not permission-settable, even by the root user. [29, 30, 31, 47, 50, 54 and 55]

File permissions asides, inodes (or their equivalent filesystem structure) are used to store the filesystem object type. For example, consider that a standard UNIX filesystem can accommodate various types of objects including files, directories, links, sockets, pipes, etc. Each of these different classes of objects can be created using the appropriate commands. For example, to create a system device consider using the *mknod* system command can be used. To create pipes, the command *mkfifo* is used; to create sockets the command *mksock* is used and to create links, the *ln* command is used. Regular files and directories can be created using many commands but they are commonly generated using the *touch* and *mkdir* commands, respectively. These commands work on all modern UNIX systems and their respective filesystems.

Each filesystem object's file type can be determined by looking at the first character of each line when carrying out a long file listing using the command *ls -l*. Knowing the specific type of file to search for, an investigator can readily seek it out[39]. Consider the following file type designations:

Table 1. Differentiating file type designation.

File listing	File type designation
-rw-rw-r--	"-" denotes ordinary file
drwxr-xr-x	"d" denotes directory
lrwxrwxrwx	"l" denotes link
prw-rw-rw	"p" denotes FIFO/pipe
srwxr-xr-x	"s" denotes socket
crw-rw-rw	"c" denotes character device
brw-rw-rw	"b" denotes block device

39 The *find* command can be used to search out specific object types.

Permissions are applied to filesystem objects as a means of granting or revoking specific authorizations to the object's owner, groups and regroupings thereof, and the public. Under UNIX, permissions are normally stored in the filesystem object's inode or equivalent structure [26, 27]. The most commonly used tools for setting and reading filesystem object permissions are the commands *chmod* and *ls*, respectively. However, to change an existing object's owner and group, the *chown* and *chgrp* commands are used, respectively. These commands apply to all modern UNIX systems and their respective UNIX filesystems.

Several other permission-like structures exist for modern UNIX filesystems and include the SUID, SGID and sticky bits, which are implemented through the *stat*() system call and are stored in the filesystem object's inode [29, 30, 31, 47, 54 and 55]. Specifically, SUID and SGID permissions are actually access right flags which enable programs associated with them to run with the privileges of either the user (SUID) or group (SGID) which set it. Sticky bits are another type of access rights flag which when set, allows users to modify files or directories which they normally would not be authorized to (e.g. */tmp*).

However, when the simple permission setting mechanisms of UNIX often do not suffice for adequately securing a system, a more fine-tuned approach is accomplished using ACLs [29, 30, 31 and 50]. ACLs use standard UNIX permissions and grant additional authorizations (*r*, *w*, *x*) to additional users, groups or the public, all of which are stored in a filesystem object's inode (or equivalent structure). Multiple ACLs can be applied to a filesystem object but different filesystems may place specific limitations as to how many are permitted for any given object.

Under Linux, ACLs are set using the *setfacl* command and can be read using the *getfacl* command. Getting a specific filesystem to work with ACLs under Linux may require it to be mounted with appropriate mount options however. Since most modern UNIX systems implement ACLs, it is important to understand how to read and set them under Linux. The reading and setting of ACLs under Linux works for some filesystems and not for others. For Extended-based filesystem, reading and setting ACLs is fully functional. Other Linux filesystems vary in their ability to support ACLs although the authors have confirmed through experimentation that Ext2/3/4, XFS, JFS and Btrfs fully support ACLs under Linux.

4 Timeline formats

4.1 Background

Although there is no industry accepted timeline format or standard, two have risen above the rest. The first of these two standards is the Bodyfile and Mactime standards which were put forward by Dan Farmer and Wietse Venema of The Coroner's Toolkit some years ago and continued on by Brian Carrier of The Sleuth Kit. The second standard, TimeLiNe (or TLN), is relatively recent and was put forward by Harlan Carvey. Both formats contrast against one another with respect to the information they convey based on the specific fields found in each of these formats. Another timeline format deserves mention as it is beginning to catch, the Common Exchange Format, or CEF.

It is important to note that while the Bodyfile is a preliminary timeline its counterpart, Mactime, is an intermediate timeline format. So to are the TLN and CEF timeline formats. Intermediate timeline formats are not only entirely text-based but are also human-readable and thus can be of use to the investigator in their current format. However, in the opinion of the authors, they are not yet suitable for use as a final timeline format.

From the point of view of this technical memorandum the ultimate objective is not only a human-readable text-based timeline but one which is legible and readily understandable. The authors accomplish this using the Bodyfile timeline format which is then converted to the Mactime format using the *mactime* tool; this format is the reformatted using scripts to prepare it for use as a finalized timeline format. This final timeline format, as proposed by the authors, is by no means ideal for all situations although the authors hope that investigators, through the use of simple scripts can reformat timeline output to suit their own specific requirements and that the authors' proposed format can serve as a basis for future custom timeline formats.

In this section several subjects are examined. The first, examined in Section 4.2.1, examines the preliminary timeline format, specifically the Bodyfile format (Section 4.2.1.1). An example is included of the Bodyfile format so that reader can better understand why this is a preliminary format and not an intermediate form (Section 4.2.1.1.1).

This is followed by an examination of various intermediate timelines formats including the TLN, CEF and the Mactime timeline formats in Section 4.2.2. However, an example timeline output is only provided for the Mactime timeline format. Example outputs for other formats are not provided herein as the authors do not consider them appropriate for filesystem-based timeline analysis.

The authors refer to the aforementioned formats as preliminary and intermediate formats because although they are comprehensible (TLN, CEF and Mactime more so than the Bodyfile format) none should be used as a final timeline format as they all are missing important identification markers such as disk image name and improved field demarcation. Examples of these identification markers can be found in the authors' proposed timeline format in Section 4.2.3.1.

The authors' proposed final timeline format, based on the Mactime timeline format, can be found in Section 4.2.3. The example output from the authors' proposed final timeline format was generated using the proposed and included prototype found in Annex B.

4.2 Examination of the various timeline formats

This section will examine the various types of timelines formats.

4.2.1 Preliminary timeline format

A preliminary timeline format, as the authors define it, is a text-based timeline which is beyond that of the raw filesystem metadata presented by various tools but which is not yet highly legible due to the need to convert the various time metadata into a more human-readable format and re-arrange the various data in order to improve its understandability. Preliminary timeline formats are generally converted to another format such as Mactime, TLN or CEF for improved legibility.

The reason the Bodyfile format is the only preliminary timeline format examined herein is because it is the only such type of timeline format in widespread use. The Bodyfile timeline format is primarily generated by The Sleuth Kit compatible filesystem metadata processing tools, although other non-compatible tools may also use a similar Bodyfile-like encoding scheme.

In general, preliminary timeline formats such as the Bodyfile cannot stand alone as they are readily legible or understandable by either the reader or the authors.

4.2.1.1 Bodyfile timeline format

The first Bodyfile format was used by The Sleuth Kit versions 1.x and 2.x and The Coroner's Toolkit both of which today are considered obsolete and should no longer be used. The second format is the newer one. It is only used by The Sleuth Kit version 3.x and other compatible tools including *log2timeline*, *ex-tip*, etc.[40]

Herein, only the more recent Bodyfile format (version 3.x) is examined as it is relevant to the various versions of The Sleuth Kit 3.x currently in use [15]. Furthermore, it is more succinct than its predecessors (versions 1.x and 2.x), which were based directly on the original Bodyfile standard put forward by Farmer and Venema [10, 11] and generated using The Coroner's Toolkit *grave-robber* tool. At any rate, the old Bodyfile timeline format was specifically designed for UNIX filesystems and metadata and was not well-suited for the more generalized approach necessary for examining non-UNIX filesystems (e.g. NTFS and FAT) [10, 11]. The newer Bodyfile format extracts only the appropriate filesystem metadata necessary to generate a meaningful digital timeline while supporting disparate filesystems.

The tools used to generate a Bodyfile from The Sleuth Kit (version 3.x) include *fls*, *ils* and *mac-robber*. The output of these tools uses the following Bodyfile format [15]:

40 See Section 2.6 for a more concise listing of which timeline generation tools support The Sleuth Kit's Bodyfile format.

```
MD5 | name | inode | mode_as_string | UID | GID | size | atime | mtime |
ctime | crtime
```

The first of the above fields is the MD5 hash of the filesystem object in question, which is not actually carried out by any of the aforementioned tools. This should be done by the investigator through the use of scripting and its usage is not required in this technical memorandum. Third-party software can be used to generate MD5 hashes and include *md5sum*, *md5deep* and others. It is useful to note that The Coroner's Toolkit Bodyfile generating tool, *grave-robber*, actually generates MD5 hashes of filesystem objects by calling an external MD5-hashing program.

Each line of output represents one specific filesystem object which is generally a file but could also be a directory, link, pipe, socket or device.

The second field is the filesystem object's name (e.g. *C:\Windows\explorer.exe*). The *inode* field is the specific filesystem inode or metadata address which stores the position of the filesystem object (recall that inodes are filesystem pointers).

The *mode_as_string* field corresponds to the UNIX permissions of the filesystem object. If the filesystem is a non-UNIX filesystem then the permissions are mapped to UNIX permissions. The object's type (file, directory, link, etc.) will also be specified in the *mode_as_string* field.

The *UID* and *GID* fields correspond to the user and group IDs of the filesystem object, respectively. While decoding *UID* and *GID* using *mactime*, it is possible to specify a password file (e.g. */etc/passwd*) for converting these values into real system user and group names. However, the use of a Windows SAM type file will not work for *UID* and *GID* conversion.

The size field corresponds to the number of bytes the filesystem object has actually been allocated on the filesystem.

The *atime*, *mtime*, *ctime* and *crtime* fields all correspond to the various MAC time filesystem-based entries for the given filesystem object, each of which is stored using UNIX Epoch time [34]. These MAC times were examined in Section 3.3. Additional information concerning various user and system-related actions and their effects on date/time metadata as well as MAC time modifications can be found in Annex A.3.

However, in order to generate a Bodyfile from the *fls* or *ils* tools, it is necessary to specify the appropriate parameters to either program. On the other hand, the *mac-robber* program does not require any parameters beyond a listing of which specific directories to scan in order to generate a Bodyfile.

4.2.1.1.1 Example Bodyfile output

An example disk image from a Windows Vista Service Pack 2 32-bit computer operating system disk image was examined using The Sleuth Kit's *fls* program. The command used to mount the disk image was as follows:

```
system$ mount -o ro,offset=1048576,loop vista32bit.dd /media/tmp
```

This command mounts the disk image *vista32bit.dd* as read-only onto mount point */media/tmp* using the Linux loop back device */dev/loop0*. The command is further instructed to perform the actual filesystem mount 1,048,576 bytes from the beginning of the disk image.

Using the following *fls* command, a Bodyfile was generated:

```
fls /dev/loop0 -p -a -r -m C: > /tmp/sample.bodyfile
```

This command instructs the *fls* program to read from */dev/loop0*, where the mounted disk image is found and display both allocated and deleted entries (-a), including full file path (-p) as well as a recursive file and directory listing (-r). The *-m C:* parameters instruct the program to write the output in Bodyfile format and append the *C:* prompt to beginning of all output.

The following output[41] will help to better understand the aforementioned *fls* command:

```
0|C:/autoexec.bat|8559-128-
1|r/rrwxrwxrwx|0|0|24|1162462989|1158615816|1300320467|1162462989

0|C:/config.sys|8560-128-
1|r/rrwxrwxrwx|0|0|10|1162464190|1158615817|1300320648|1162448708

0|C:/Documents and Settings|8563-144-1|d/dr-xr-xr-
x|0|0|48|1162472544|1162472544|1293486419|1162472544

0|C:/Documents and Settings/.|8563-144-1|d/dr-xr-xr-
x|0|0|48|1162472544|1162472544|1293486419|1162472544

0|C:/Documents and Settings/..|5-144-6|d/dr-xr-xr-
x|0|0|160|1300326695|1300326695|1300326695|1162462736

0|C:/PerfLogs|60-144-
1|d/drwxrwxrwx|0|0|144|1200882790|1200882790|1300320470|1200882790

0|C:/PerfLogs/.|60-144-
1|d/drwxrwxrwx|0|0|144|1200882790|1200882790|1300320470|1200882790

0|C:/PerfLogs/..|5-144-6|d/dr-xr-xr-
x|0|0|160|1300326695|1300326695|1300326695|1162462736

0|C:/PerfLogs/Admin|61-144-
1|d/drwxrwxrwx|0|0|48|1200882790|1200882790|1293486410|1200882790

0|C:/PerfLogs/Admin/.|61-144-
1|d/drwxrwxrwx|0|0|48|1200882790|1200882790|1293486410|1200882790

0|C:/PerfLogs/Admin/..|60-144-
1|d/drwxrwxrwx|0|0|144|1200882790|1200882790|1300320470|1200882790
```

41 Line spacing inserted by authors to improve visibility.

```
0|C:/Program Files|62-144-6|d/d-wx-wx-wx|0|0|280|1300326865|1300326865|1300326865|1162466313

0|C:/Program Files/.|62-144-6|d/d-wx-wx-wx|0|0|280|1300326865|1300326865|1300326865|1162466313

0|C:/Program Files/..|5-144-6|d/dr-xr-xr-x|0|0|160|1300326695|1300326695|1300326695|1162462736

0|C:/Program Files/Adobe|32970-144-1|d/drwxrwxrwx|0|0|256|1293477682|1293477682|1300320470|1293477682

0|C:/Program Files/Adobe/.|32970-144-1|d/drwxrwxrwx|0|0|256|1293477682|1293477682|1300320470|1293477682
```

4.2.2 Intermediate timeline formats

Intermediate timeline formats, as the authors have defined them, are text-based human-readable timelines which are far improved over more basic timelines such as the Bodyfile timeline format. Most timeline formats do not have precursor timeline formats such as the Bodyfile format. Moreover, unlike the Bodyfile format which relies on tools such as *mactime* and *log2timeline*[42] to convert it into a more readable and useful format, intermediate formats including TLN and CEF are also considered as final timeline formats. The same can be said for the Mactime format, although the authors believe that all intermediate formats require additional processing to enable them to be even more useful and immediately understandable, even by those not fluent in digital forensics or computer timeline formats. As such, even though the intermediate formats can stand on their own they can all use improvement.

Consider that for the TLN timeline format, tools such as *log2timeline* can generate timelines directly into this format from filesystem metadata and other additional metadata sources fed to it rather than rely on a transitional format such as the Bodyfile timeline format. The same is true for the CEF timeline format. However, for reasons to be examined below, it will become clearer why, at least in the opinion of the authors, why these formats should not be used as finalized timeline formats even if they can stand on their own.

As for the Bodyfile timeline format, many tools encode directly into this format whereas others instead encode directly into the Mactime timeline format. The Sleuth Kit, for example, which is an essential component of filesystem timeline generation for this technical memorandum, encodes filesystem metadata directly into the Bodyfile timeline format and later requires the *mactime* tool in order to convert from this format to the Mactime format, which is more comprehensible than its predecessor.

However, tools such as *log2timeline* change things. Not only can they read and parse the Bodyfile timeline format but it can also export it and other timeline formats to and from other formats including but not limited to the TLN, Mactime and CEF timeline formats.

---

[42] *Log2timeline* can parse and export the Bodyfile format to other formats including TLN and CEF, to name a few.

In general, all intermediate timeline formats are legible and understandable, whether or not the readers or the authors consider them to be finalized timeline formats.

4.2.2.1 TLN timeline format

In comparison to the Bodyfile format, the TLN format proposed by Harlan Carvey is simple in comparison [79, 80]. TLN is a format which is specific to the tools and utilities written by Harlan Carvey (e.g. *regripper,* etc.). The present authors propound that Carvey defined his own timeline format to make the output from his tools more efficient as the Bodyfile format can be cumbersome to work with [79, 80]. The TLN format is defined as [79, 80]:

```
Time | Source | Host | User | Description
```

Time is stored in either UNIX Epoch time or Windows 64-bit time and its use will depend on the data source (e.g. Windows registry uses Windows 64-bit time while UNIX filesystem object uses 32-bit UNIX time). The *source* is defined as a brief description of the information source (e.g. registry, filesystem, EVT/EVTX files, etc.). The *host* typically defines some information about the system where the information was found (e.g. system name, DNS name, IP address, MAC address, etc.). The *user* is a string providing some description of the user account under which the information source resided (e.g. system username, Windows SID, e-mail address, etc.). Finally, the *description* is a short concise statement of what happened or what was done. [79, 80]

It is important to note that according to Harvey, not all the fields will be filled at all times. The absence of certain information will depend in part on the data source and the program generating the timeline. [79, 80]

Although the TLN format is an attractive option in comparison to the Bodyfile format or even the Mactime timeline format, in the opinion of the authors, it is somewhat lacking in order to be truly useful in generating forensic timelines. As such, there is little else to examine concerning the TLN format.

4.2.2.2 Mactime timeline format

Although both The Coroner's Toolkit and The Sleuth Kit include the *mactime* tool, they perform different functions under each suite. The Coroner's Toolkit's *mactime* tool performs the same task as the *mac-robber* tool. On the other hand, The Sleuth Kit's *mactime* tool converts Bodyfiles to human-readable text-based timelines which are commonly referred to as Mactime timelines.

Once a Bodyfile output has been generated, The Sleuth Kit's *mactime* tool is used to convert it to the Mactime format by consolidating the various *atime*, *ctime*, *mtime* and *crtime* metadata into something more meaningful and convenient for the investigator by providing an easy to read date/time output. Output from the *mactime* tool can vary depending on the various command line parameters fed to the program.

By default, the *mactime* program will group all filesystem objects found with the same time together. However, since date and time can be obtained from atime, ctime, mtime or crtime fields, *mactime* will regroup objects according to the same date/time regardless of which field that

date/time is from. Looking at the example default Mactime format output in Section 4.3.2.1 will clarify the matter.

By default, the output of the *mactime* tool has the following format [17]:

```
Date | Size | Type | Mode | UID | GID | Meta | File Name
```

Many of the fields in the Mactime format output are similar to those found used for the Bodyfile format. However, several require explanation. The field *Type* does not refer to file type but rather refers to the *activity type* or MAC time (see Section 3.3 for more details). The other item requiring precision is the field *Meta* which refers to the *metadata address* or *inode* of the filesystem object in question. [17]

An example of this output is found in the ensuing section below.

4.2.2.2.1 Example of a Mactime timeline output

An example of the default output from the *mactime* program once it has read in a specified Bodyfile and generated a human-readable text-based timeline is shown below and was generated using the following command:

```
system$ mactime -b bodyfile.txt
```

Generates the following *mactime* timeline output[43]:

```
Tue Aug 19 1997 02:37:00     15558 m..b r/rrwxrwxrwx 0        0        62452-128-3
C:/Program Files/Common Files/microsoft shared/GRPHFLT/MS.CDR

                          6615 m..b r/rrwxrwxrwx 0       0       62453-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/CGMimp32.CFG

                        606062 m..b r/rrwxrwxrwx 0       0       62454-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/CGMimp32.FNT

                          1908 m..b r/rrwxrwxrwx 0       0       62455-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/MS.CGM

                          1069 m..b r/rrwxrwxrwx 0       0       62456-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/MS.GIF

                          1061 m..b r/rrwxrwxrwx 0       0       62457-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/MS.JPG
```

43 Line spacing inserted by authors to improve visibility.

```
                    1682 m..b r/rrwxrwxrwx 0       0       62458-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/MS.PNG

                    1382 m..b r/rrwxrwxrwx 0       0       62459-128-3

C:/Program Files/Common Files/microsoft shared/GRPHFLT/MS.WPG

                   15067 m..b r/r-wx-wx-wx 0       0       62467-128-3

        ...

Wed Jun 24 1998 00:00:00    137000 m... r/rrwxrwxrwx 0          0          33395-128-3
C:/Windows/System32/MSMAPI32.OCX

Mon Jul 06 1998 00:00:00     23552 m... r/rrwxrwxrwx 0          0          33145-128-3
C:/Windows/System32/MSMPIDE.DLL

Tue Jan 26 1999 03:07:36        26 m... r/rrwxrwxrwx 0          0          56164-128-1
C:/Program Files/Jetico/BCWipe/BCVIEW.INI

Wed Apr 14 1999 15:46:54     34705 m..b r/r-wx-wx-wx 0          0          56991-128-3
C:/Program Files/Adobe/Reader 9.0/Resource/Font/SY______.PFB

                   75573 m..b r/r-wx-wx-wx 0       0       56992-128-3

C:/Program Files/Adobe/Reader 9.0/Resource/Font/ZX______.PFB

                   96418 m..b r/r-wx-wx-wx 0       0       56993-128-3

C:/Program Files/Adobe/Reader 9.0/Resource/Font/ZY______.PFB

                     672 m..b r/r-wx-wx-wx 0       0       56995-128-1

C:/Program Files/Adobe/Reader 9.0/Resource/Font/PFM/SY______.PFM

                     683 m..b r/r-wx-wx-wx 0       0       56996-128-1

C:/Program Files/Adobe/Reader 9.0/Resource/Font/PFM/zx______.pfm

                     684 m..b r/r-wx-wx-wx 0       0       56997-128-1

C:/Program Files/Adobe/Reader 9.0/Resource/Font/PFM/zy______.pfm
```

As can be seen from this example, filesystem objects are grouped according to their date/time. This feature can be useful but it is a significant impediment when attempting to perform *grep*-based searches against the timeline.

However, generating Mactime timelines using different command line parameters can have the effect of date/time stamping each entry such that they can be more readily examined for patterns using *grep* (or another similar tool).

4.2.2.3 Other potential timeline formats

Although one or more of the following potential timelines could have been chosen for the work and prototype proposed herein over the Mactime timeline format, since this work is primarily concerned with disk-based Windows images broader scope timeline formats such as CEF are not particularly applicable. The other possible timeline formats have no particular merits over the Mactime timeline format

4.2.2.3.1 CEF – the Common Exchange Format timeline format

This format is a relatively unknown newcomer to the timeline format space. The Common Exchange Format or CEF as it is known by, was proposed by ArcSight Inc. Its format is not particularly different from that of the TLN or Mactime timeline formats. However, it supports many more field types than either of the aforementioned formats. And although it was designed for the enterprise management of log and event generating software and devices such as AV command consoles, IDS and NIDS systems, it could also be used for generating filesystem-based timelines. However, this format has far too many fields to describe herein in a concise manner and as such it is not examined in-depth in this work. [83, 84]

It is the opinion of the authors that the CEF timeline format is best left for use with log and event generating systems and devices. This format can and should be used for timeline analysis of network based events and data garnered from data sources including PCAP-based network logs, Squid network logs, Windows firewall logs, Windows ISA Proxy logs, Apache web server logs and Microsoft IIS logs, all of which can be processed by *log2timeline* and exported to another format. [66, 68, 83 and 84]

4.2.2.3.2 TLNX – TLX XML

This timeline format is an XML representation of the TLN timeline format. This format is not currently in high use and as such it is not examined in this work.

4.2.2.3.3 CSV – Comma Separated Values

This timeline format can be readily generated by *log2timeline* but it can also be generated by the *mactime* tool using the *-d* command line option. CSV data files are especially used for importing data into spreadsheets and databases and as such may be of use to investigators requiring the important of timeline specific data which is to be stored within such a system. However, this format due to its simplicity, is very easy to understand upon visual inspection of a given CSV data file, which is why it is not examined in this work.

4.2.2.3.4 SIMILE

The SIMILE timeline format is an XML representation of timeline data which is for the exclusive use of the SIMILE *timeplot* widget. Refer to Section 2.6 for more details.

4.2.3 Authors' proposed enhanced Mactime timeline format for use as a final timeline layout

Although the authors could have chosen to work with any of the aforementioned timeline formats they have chosen to concentrate their efforts on the Mactime timeline format because it is, in the opinion of the authors, not necessarily the superior format but certainly one of the most commonly used (if not the most commonly used[44]). Moreover, because The Sleuth Kit is the premier open source-based filesystem analysis tool, a primary objective of the authors was to remain as close as possible to the standards set for by The Sleuth Kit in order to progress their own respective works. As such, with an aspiration to share their work and in particular their own take on the Mactime timeline format with a fully functional timeline generation prototype have undertaken it upon themselves to write this paper and share it with the digital forensics community. However, the author's proposed enhanced timeline also makes generous use of the *log2timeline* timelines analysis generation software suite due to its expanded data file import, log analysis, and exportation capabilities.

Although disk image labelling may not be of importance when generating a timeline analysis of one or several disk images, it is of immense help when working with many disk images. This becomes all the more significant when performing pattern or date/time distribution analysis (among many other types of analyses which can potentially be carried out) against numerous disk images. For this reason, the authors have proposed that each timeline entry be preceded by a disk image name as is found in Section 4.2.3.2.

4.2.3.1 Proposed timeline format specifics

As all Mactime timeline format output is text-based and field delimited, it is possible to reformat it into something more meaningful. As such, in the context of this technical memorandum, the *mactime* program was instructed to generate a CSV-based timeline which could then be appropriately formatted in order to emphasize readability. Moreover, the *mactime* program was further instructed to convert all time output to EST5EDT which is the time zone from whence came various test suspect systems originated.

The above *mactime* timeline output is then piped multiple times into the *sed* text data processing utility to remove all non-filesystem object name commas (",") and replace them with pipes ("|") in order to improve readability and field demarcation.

Thus, while the author's enhanced Mactime timeline output closely resembles that of the default Mactime timeline output (see Section 4.2.2.2), they are nonetheless different. The first difference is that each listed object is preceded by the name of the disk image from whence it originated. This is then followed by each object being date/time stamped rather than regrouped by a single date/time stamp. The third is that all date/time output is no longer based on local time but on a user specified time zone[45]. Finally, the field delimiter is no longer the tab character (\t) and is now the "|" character preceded and followed by several spaces to improve readability. The content itself, however, remains altogether unchanged.

---

44 Timeline format usage-based statistics are not available at this time from any known source.

45 This must be a valid time zone.

4.2.3.2 Example of enhanced Mactime timeline format

The following is an example of the enhanced Mactime timeline output. However, this is based solely on the preferences of the authors and the reader is free to use an altogether different formatting. Moreover, reformatting is made simple and uniform by using standard UNIX text and processing tools such as *sed* and *awk*.

These *awk* and *sed* commands can be found in the timeline control script in Annex B.1. As such, using these commands a Mactime-based timeline format has been reformatted into a more legible format as follows below:

```
Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 15558 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62452-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/MS.CDR"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 6615 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62453-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/CGMimp32.CFG"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 606062 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62454-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/CGMimp32.FNT"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 1908 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62455-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/MS.CGM"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 1069 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62456-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/MS.GIF"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 1061 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62457-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/MS.JPG"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 1682 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62458-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/MS.PNG"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 1382 | m..b |
r/rrwxrwxrwx | 0 | 0 | 62459-128-3 | "C:/Program Files/Common
Files/microsoft shared/GRPHFLT/MS.WPG"

Exploited_PC_disk_image--> Tue Aug 19 1997 02:37:00 | 15067 | m..b | r/r-
wx-wx-wx | 0 | 0 | 62467-128-3 | "C:/Windows/Installer/$PatchCache
```

In comparison to the default Mactime timeline format presented in Section 4.2.2.2 the authors are of the opinion that this reformatted output is much improved. The "|" characters, in the opinion of the authors, serve to improve readability over the default tab character (\t).

4.3 Reading and processing Mactime-based timeline output

Reading and processing Mactime-based timeline is not difficult because it is text-based and field delimited. Its legibility can be further improved through the use of customized formatting by using additional text processing filters and processors (e.g. *sed*, *awk*, *grep*, etc.). When reading a standard Mactime timeline output, the two most important items to search for are the date/time and activity type. These two items will allow the investigator to quickly narrow in on specific types of file activity which occurred at key times of interest.

In so doing, the investigator must be conscientious that filesystem objects are under constant modification so long as the operating system is operational and the user is accessing and modifying both system[46] and data files. Files constantly undergo changes in name, size, contents and volume location due to a myriad of reasons such as file updates, service packs, user-induced data file changes, etc. These changes will affect the date/time metadata (*atime*, *ctime*, *mtime* and *crtime*[47]) of the effected file(s). It is important to recall that the various date/time metadata are modified whenever a change is made to the file or its metadata (see Section 3.3 for more details).

Thus, over time, it is expected that many filesystem objects will experience normal system and user-based changes, all of which are reflected in their date/time metadata. For example, consider the following example output[48] which examines the changes which have occurred to the system file *C:\autoexec.bat*:

```
Mon Sep 18 2006 17:43:36 | 24 | m... | r/rrwxrwxrwx | 0 | 0 | 8559-
128-1 | "C:/autoexec.bat"

Thu Nov 02 2006 05:23:09 | 24 | .a.b | r/rrwxrwxrwx | 0 | 0 | 8559-
128-1 | "C:/autoexec.bat"

Wed Mar 16 2011 20:07:47 | 24 | ..c. | r/rrwxrwxrwx | 0 | 0 | 8559-
128-1 | "C:/autoexec.bat"
```

Understanding the changes that were made to the example file above are not necessarily straightforward. Different types of file activity will result in varying date/time metadata changes being recorded. As such, the connection between file activity and date/time metadata changes are readily perceived. However, in order to attempt to identify which file activities resulted in the recorded date/time metadata change sought after by the investigator, some investigative work may be required. It is useful to recall that all file activities (under NTFS there are four MAC times) are recorded in the date/time metadata of the filesystem object in question. Using the information prepared in Annex A.3 which is up to date, the investigator can attempt to recreate events leading up to specific file activities and date/time changes.

---

46 The user may be able to modify system files only if he has permissions to do so or if permissions are enforced by the operating system. Even if the user cannot modify system files, his actions while using the system are likely recorded by the system event logger (and system auditor, if enabled).

47 NTFS only metadata.

48 Line spacing inserted by authors to improve visibility.

Unfortunately, in many cases, disparate file activities will have the same end result as far as date/time metadata changes are concerned. Thus, at times the investigator cannot be absolutely certain that a given set of file changes resulted in the associated date/time metadata changes. There is nothing that can be done about this due to the use of current filesystem data structures necessary, which despite their deficiencies are required for recording various date/time metadata activities. If only additional filesystem metadata structures existed which could be used to accommodate the plethora of present filesystem object-based activities then it would be possible to discern with far greater precision which actions led to the current state of said metadata structures.

Regrettably, no known filesystem actually implements sufficient filesystem metadata structures. Even NTFS with its vast array of MFT metadata structures still can still leave the investigator pondering which activity led to the present date/time metadata values found in said structures (see Annex A.3 for more details).

Further hampering the situation is the use of the currently accepted MAC times (a.k.a. MACB times). Currently, MAC time support for all filesystems is limited to *atime*, *ctime*, and *mtime*, with the notable exceptions of NTFS and Ext2/3/4. NTFS supports *atime*, *ctime*, *mtime* and *crtime* [21, 25, 28, 42, 45, 46 and 53] while Ext2/3/4 supports *atime*, *ctime*, *mtime* and *dtime* [20, 24, 28, 29, 30, 31, 32, 33 and 41]. However, The Sleuth Kit (and all other filesystem-based timeline tools) does not currently provide MAC support for *dtime*, only NTFS' *crtime*.

Consider this final example output[49]. The file *C:\Windows\System32\ntdll.dll*, a very important system file for Windows operating systems, has undergone at least the following file activity changes from a test Windows Vista operating system disk image.

```
Sun Jan 20 2008 21:25:27 | 1203792 | ma.b | r/rrwxrwxrwx | 0 | 0 |
17730-128-3 | "C:/Windows/System32/ntdll.dll (deleted-realloc)"

Fri Oct 15 2010 09:48:59 | 1205080 | m... | r/rrwxrwxrwx | 0 | 0 |
50827-128-4 | "C:/Windows/System32/ntdll.dll"

Mon Dec 27 2010 14:07:01 | 1203792 | ..c. | r/rrwxrwxrwx | 0 | 0 |
17730-128-3 | "C:/Windows/System32/ntdll.dll (deleted-realloc)"

Sat Feb 12 2011 14:02:44 | 1205080 | .a.b | r/rrwxrwxrwx | 0 | 0 |
50827-128-4 | "C:/Windows/System32/ntdll.dll"

Sat Feb 12 2011 15:50:45 | 1205080 | ..c. | r/rrwxrwxrwx | 0 | 0 |
50827-128-4 | "C:/Windows/System32/ntdll.dll"
```

Attempting to recreate the file activities which resulted in the date/time metadata changes being recorded is even more challenging than for the file activities associated with the file *C:\autoexec.bat*.

49 Line spacing inserted by authors to improve visibility.

4.4 A final point about UNIX data processing

UNIX text and data processing utilities can be used to not only better format the mactime timeline output but can be used to pull out specific dates, names, permissions, file types, file activity, etc. Using the *grep* utility, it will be possible to perform multiple word and pattern searches in one pass. The *sort* utility can be used directly on the timeline or its pre-processed output from another tool (e.g. *grep*), all of which can be sorted according to various criteria. The *uniq* tool is generally used to remove duplicate lines but it can also be used to add up the number of times a given keyword or date occurs in an output.

Another very powerful scripting tool is the *awk* utility which can be used to not only reformat timelines and other output, but through *awk* programming, the investigator can create self-contained programs to perform additional work and data processing including pattern searching and analysis. The final text and data processing tool which the authors find to be of use when working with timelines is the *tr* command, which is a character translation tool and can replace one or more characters by another set of one or more characters and is useful for troublesome filename characters and field delimiters (e.g. "$", "@", etc.).

All these capabilities may take some time for the investigator to adjust to, but can be of immense value, particularly when dealing with very large timeline analyses.

5 Examining timeline-based sources of information

5.1 Background

In this section, the sources of date/time metadata available for inclusion in a digital timeline by the software tool *log2timeline* are examined as are the actual sources of data used by the authors in their digital timeline analysis implementation (see Annex B.1 to examine the control script in more detail).

5.2 Sources of date/time metadata

5.2.1 Sources available through log2timeline

According to [66], the current version of *log2timeline,* version 0.52, supports the following sources of date/time metadata:

- Apache2 access logs
- Apache2 error logs
- Google Chrome history
- Encase directory listings
- Windows event log files (EVT)
- Windows event log files (EVTX)
- EXIF information or metadata from various media files
- Firefox bookmarks
- Firefox 2 history
- Firefox 3 history
- FTK Imager directory listing CSV files
- Generic Linux log file
- Internet Explorer history files, parsing *index.dat* files
- Windows IIS W3C log files
- ISA server text export (copy query results to clipboard and into a text file)
- *Mactime* Bodyfiles (to provide an easy method to modify from *mactime* format to some other)
- McAfee AntiVirus log files
- MS-SQL error logs
- Opera global and direct browser history
- OpenXML metadata (for metadata extraction from Office 2007 and 2010 documents)
- PCAP files (parsing network dump files created by tool such as *Wireshark* and *Tcpdump*)
- PDF (parse the basic PDF metadata to capture creation dates, etc. from PDF documents)
- Windows Prefetch directory
- Windows Recycle Bin (INFO2 or I$)
- Windows Restore Points
- Safari Browser history files

- Windows XP SetupAPI log files
- Adobe Local Shared Object files (SOL/LSO) (a.k.a. Flash Cookies)
- Squid Access Logs (httpd_emulate off)
- TLN (timeline) Bodyfiles
- UserAssist key of the Windows registry
- Volatility (based on the output file from the *psscan* and *psscan2* modules from volatility)
- Windows Shortcut files (LNK)
- Windows WMIProv log files
- Windows XP Firewall Log files (W3C format)

And it supports the following timeline formats [66]:

- BeeDocs
- CEF
- CFTL
- CSV
- Mactime
- SIMILE
- SQLite
- TLN (CSV and tab ("\t") delimited formats both supported)

Although a typical Windows operating system disk contains a multitude of date/time metadata, not all sources are necessarily useful. Incorporating too many sources of information may cause the investigator to experience information overload. This risk becomes all the more important as the number of metadata sources increase. It is for this reason that the investigator, in the opinion of the authors, must plan ahead of time with respect to which metadata sources are considered important to his investigation.

Moreover, although it may seem appropriate to extract and include all EXIF information wherever found, this is generally not a good idea, again mainly due to potential of information overload. If geo-location information[50] is required then EXIF metadata extraction is a must. [82]

Many of the aforementioned data sources are likely not pertinent to a specific investigation and as such should not be considered for inclusion in a digital timeline.

The one problem with "one-stop-shop" programs such as *log2timeline* is that they require that the investigator specify a listing a files to extract date/time metadata from. This requires that investigator know where these files reside or have the ability to find these files. This is where programs such as the *find* and *file* command become very useful.

[50] Cameras and cell phones can store geo-location information in images and videos. Other data types likely exist.

5.2.2 Sources used and implemented in the proposed timeline extraction framework

As implemented by the control script written by the authors, the timeline extraction and analysis framework (see Annex B.1), searches a given raw disk image for some file types using the *find* and *file* commands. Other file types which are known either by specific filename or location are instead fed directly to the *log2timeline* program. And in other cases, where the filename, location or *file* command-based signature is unknown, author-provided signature detection software are used to locate the desired files and feed them to *log2timeline*.

Thus, the author-provided prototype script (see Annex B.1) relies not only on various shell commands, but The Sleuth Kit, *log2timeline* and an author-provided and developed signature detection program (see Annex B.2.2). The various shell commands used by the script include but are not limited to *file*, *find*, *tr*, *cpio*, *sed*, *grep* and *awk*.

As carried out herein, based on the abilities of The Sleuth Kit, *log2timeline* and the specific needs of the authors concerning their own investigations, the following sources of time-based metadata are extracted from a given disk image which is carried out by the proposed prototype in Annex B:

- Allocated filesystem objects
  → Implemented using The Sleuth Kit
- Deleted filesystem objects
  → Implemented through The Sleuth Kit
- Undeletable filesystem objects
  → Implemented using The Sleuth Kit
- Windows registry objects
  → Implemented using Harlan Carvey's *regtime.pl* Perl script
- Windows event logs (EVT[51] and EVTX formats)
  → Implemented using log2timeline
- Windows prefetch files
  → Implemented using *log2timeline*
- Windows system restores
  → Implemented using *log2timeline*
- Windows shortcuts
  → Implemented using log2timeline
- Windows Internet Explorer history files
  → Implemented using *log2timeline*
- Firefox 3[52] history files

[51] The control script, which uses the *file* command to find the various data types which are fed to *log2timeline* cannot detect Windows EVT logs (Windows XP, 2000, and NT) and as such requires the use of an byte signature detection program, *find_eventlog_signature.c* (see Annex B.2.2 for more information), to find them.

→ Implemented using *log2timeline*

- Windows Setupapi logs
  → Implemented using *log2timeline*
- Windows firewall logs
  → Implemented using *log2timeline*
- Flash cookies
  → Implemented using *log2timeline*

5.3 How the control script works

The control script, *timeline.sh*, as seen in Annex B.1 is relatively straightforward. The script requires that the user provide as input five distinct items on the command line when initiating the script in order for it to begin processing a disk image for timeline analysis. These items, in their correct order, are:

- Timeline repository location
- Filesystem offset
- Disk image or evidence file name
- Mount point
- Partition number

The first item specifies an existing directory where all timeline files, generated or processed, are to be stored. The existing directory has a new directory appended to it, *timeline/*. The filesystem offset specifies how many bytes the actual filesystem is from the partition table which defines it. The disk image or evidence file name is the actual raw disk image which contains the filesystem(s) to be analysed. This image must be uncompressed and in a raw format as it will not process other formats[53]. Once the disk image has been found to exist, it is then mounted on the command line specified mount point. Finally, the user specifies which partition is to be worked on by the script so that it can append the appropriate partition number to timeline repository. Once complete, the processing then begins.

The first data processing requires The Sleuth Kit. All allocated disk image filesystem objects are analysed. Then all deleted and undeletable files are analysed. All appropriate filesystem metadata is then in stored using the Bodyfile timeline format which is then converted to the Mactime timeline format.

Once complete, various Windows-specific file types are sought out, including but not limited to the Windows registry, event logs, etc (see Section 5.2.2 for more details). Only the Windows EVT logs cannot be readily found. Instead, the use of a specific byte signature detection program is required.

---

[52] Firefox is currently at version 6.0. All versions as of version 3.0 use a SQLite database for storing browsing history and related information. Although *log2timeline* supports Firefox version 2.x web browsing history it is not support by the proposed timeline extraction framework since it is very outdated.

[53] Non-raw formats may include EnCase's disk format, the AFF disk format and others.

Then some basic UNIX text and data processing is carried to concatenate all the variously extracted date/time timeline data files into one final timeline data file. Each timeline entry is then preceded by the disk image name from whence it came so that when the investigator combines together all the final timelines from all the various disk images together pattern and distribution analysis will be highly simplified when discerning which event occurred on what filesystem and disk.

Finally, the disk image is unmounted and the script terminates. If there are multiple disk images or partitions to analyse the investigator has only to provide the appropriate information on the command line. The investigator could even use another script which provides all the necessary information to the control script in order to instantiate one or more timeline processing instances. Very powerful workstations with high disk I/O could even instantiate simultaneous control script instances of multiple disk images.

5.4 Sample output

In this section, the various date/time metadata sources used by the authors' control script (see Annex B.1), upon having been converted to suitable final timeline format, are examined in the subsections below.

5.4.1 Allocated filesystem objects

The following is a sample of a final timeline output for various allocated filesystem objects:

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@statsdlg/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@subsysDataLogsNode/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@timedata/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@timeFromWorkspaceDlg/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@timeplot/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@timereset/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@timeview/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@transaction/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tsCharLineView/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tsCharVarView/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tscollectionNode/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tscorrnode/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tshistnode/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tsImportdlg/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tsMeanData/.."

exploited.dd--> 2006 04 26 Wed 15:29:23 | 56 | m.c. | d/drwxrwxrwx | 0 | 0 | 249833-144-6 | "/Program Files/MATLAB/R2006a/toolbox/matlab/timeseries @tsguis/@tsMeanView/.."

5.4.2 Deleted filesystem objects

The following is a sample of a final timeline output for various deleted filesystem objects:

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714521-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[39].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714522-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG bigweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[3].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714523-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[4].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714525-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[6].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714526-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[7].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714527-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[8].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 10 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714528-128-1 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/asbs111[9].js (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 11514 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714529-128-4 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ astro_r02_c11[1].jpg (deleted)"

exploited.dd--> 2008 09 29 Mon 09:40:05 | 12129 | .ac. | -/rrwxrwxrwx | 0 | 0 | 714530-128-4 | "/weiner/oldcomputer/Documents and Settings/bigweiner.TBG/ igweiner/Local Settings/Temporary Internet Files/Content.IE5/UJYVY1IJ/ stro_r03_c01[1].jpg (deleted)"

5.4.3 Undeletable filesystem objects

The following is a sample of a final timeline output for various undeletable filesystem objects:

exploited.dd--> 2009 07 06 Mon 13:31:31 | 3669 | ...b | r/rrwxrwxrwx | 0 | 0 | 982324-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/ SP.NETWebAdminFiles/AppConfig/CreateAppSetting.aspx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 12253 | ...b | r/rrwxrwxrwx | 0 | 0 | 982325-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/DebugAndTrace.aspx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 2304 | ...b | r/rrwxrwxrwx | 0 | 0 | 982327-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/EditAppSetting.aspx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 15196 | ...b | r/rrwxrwxrwx | 0 | 0 | 982328-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/ManageAppSettings.aspx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 17787 | ...b | r/rrwxrwxrwx | 0 | 0 | 982329-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/SmtpSettings.aspx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 56 | m.cb | d/drwxrwxrwx | 0 | 0 | 982330-144-6 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/App_LocalResources"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 3806 | ...b | r/rrwxrwxrwx | 0 | 0 | 982331-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/App_LocalResources/AppConfigHome.aspx.resx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 1367 | ...b | r/rrwxrwxrwx | 0 | 0 | 982332-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/App_LocalResources/AppSetting.ascx.resx"

exploited.dd--> 2009 07 06 Mon 13:31:31 | 1539 | ...b | r/rrwxrwxrwx | 0 | 0 | 982333-128-4 | "/BUBB_A/WINDOWS/Microsoft.NET/Framework/v2.0.50727/SP.NETWebAdminFiles/AppConfig/App_LocalResources/CreateAppSetting.aspx.resx"

5.4.4 Windows registry objects

The following is a sample of a final timeline output for various Windows registry objects:

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/PCIIDE/IDEChannel/4&2c5d1230&0&1 "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/Root/dmio/0000 "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/Root/ftdisk/0000 "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Services/Cdrom "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Services/Disk "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Services/Imapi "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Services/atapi "

exploited.dd--> 2010 05 26 Wed 15:07:46 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Services/redbook "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Control/DeviceClasses/{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}/##?#STORAGE#Volume#1&30a96598&0&Signature2C9EEOffset7E00Length1C9F7F4600#{53f5630d-b6bf-11d0-94f2-00a0c91efb8b} "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Control/DeviceClasses/{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}/##?#STORAGE#Volume#1&30a96598&0&Signature2C9EEOffset7E00Length1C9F7F4600#{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}/# "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Control/DeviceClasses/{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}/##?#STORAGE#Volume#1&30a96598&0&Signature510D510COffset7E00Length2542978200#{53f5630d-b6bf-11d0-94f2-00a0c91efb8b} "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Control/DeviceClasses/{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}/##?#STORAGE#Volume#1&30a96598&0&Signature510D510COffset7E00Length2542978200#{53f5630d-b6bf-11d0-94f2-00a0c91efb8b}/# "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/Root/LEGACY_FLTMGR/0000 "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/Root/LEGACY_MUP/0000 "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/Root/LEGACY_NTFS/0000 "

exploited.dd--> 2010 05 26 Wed 15:07:47 | 0 | m... | 0 | 0 | 0 | 0 | ".//ControlSet001/Enum/Root/LEGACY_SR/0000 "

5.4.5 Windows event logs

The following is a sample of a final timeline output for various Windows event logs:

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

exploited.dd--> 2005 12 12 Wed 19:00:00 copyright SID:FruitLoop
Message: YouHoo Research | LLC

5.4.6 Windows prefetch files

The following is a sample of a final timeline output for various Windows prefetch files:

exploited.dd--> 2010 05 05 Wed 10:56:24 | 0 | macb | 0 | 0 | 0 | 237978 | "[XP Prefetch] (Last run) IEXPLORE.EXE-2D97EBE6.pf - [IEXPLORE.EXE] was executed - run count [1510]- full path: [C:/PROGRAM FILES/INTERNET EXPLORER/IEXPLORE.EXE] - DLLs loaded: {WINDOWS/SYSTEM32/NTDLL.DLL - WINDOWS/SYSTEM32/KERNEL32.DLL - WINDOWS/SYSTEM32/MSVCRT.DLL - WINDOWS/SYSTEM32/ADVAPI32.DLL - WINDOWS/SYSTEM32/RPCRT4.DLL - WINDOWS/SYSTEM32/SECUR32.DLL - WINDOWS/SYSTEM32/USER32.DLL - WINDOWS/SYSTEM32/GDI32.DLL - WINDOWS/SYSTEM32/OLE32.DLL - WINDOWS/SYSTEM32/SHIMENG.DLL - WINDOWS/APPPATCH/ACGENRAL.DLL - WINDOWS/SYSTEM32/WINMM.DLL

- WINDOWS/SYSTEM32/OLEAUT32.DLL - WINDOWS/SYSTEM32/MSACM32.DLL - WINDOWS/SYSTEM32/VERSION.DLL - WINDOWS/SYSTEM32/SHELL32.DLL - WINDOWS/SYSTEM32/SHLWAPI.DLL - WINDOWS/SYSTEM32/USERENV.DLL - WINDOWS/SYSTEM32/UXTHEME.DLL - WINDOWS/SYSTEM32/IMM32.DLL - WINDOWS/WINSXS/X86_MICROSOFT.WINDOWS.COMMON-CONTROLS_6595B64144CCF1DF_6.0.2600.5512_X-WW_35D4CE83/COMCTL32.DLL - WINDOWS/SYSTEM32/COMCTL32.DLL - WINDOWS/SYSTEM32/RPCSS.DLL - WINDOWS/SYSTEM32/DFRGRES.DLL - WINDOWS/SYSTEM32/NTMARTA.DLL - WINDOWS/SYSTEM32/SAMLIB.DLL - WINDOWS/SYSTEM32/WLDAP32.DLL - WINDOWS/SYSTEM32/CLBCATQ.DLL - WINDOWS/SYSTEM32/COMRES.DLL - WINDOWS/SYSTEM32/XPSP2RES.DLL - WINDOWS/SYSTEM32/WINSTA.DLL - WINDOWS/SYSTEM32/NETAPI32.DLL - WINDOWS/SYSTEM32/RDPSND.DLL - WINDOWS/SYSTEM32/PSAPI.DLLWINDOWS/SYSTEM32/NTDLL.DLL - WINDOWS/SYSTEM32/KERNEL32.DLL - WINDOWS/SYSTEM32/ADVAPI32.DLL - WINDOWS/SYSTEM32/RPCRT4.DLL - WINDOWS/SYSTEM32/SECUR32.DLL - WINDOWS/SYSTEM32/USER32.DLL - WINDOWS/SYSTEM32/GDI32.DLL - WINDOWS/SYSTEM32/OLE32.DLL - WINDOWS/SYSTEM32/MSVCRT.DLL - WINDOWS/SYSTEM32/OLEAUT32.DLL - WINDOWS/SYSTEM32/SETUPAPI.DLL - WINDOWS/SYSTEM32/SHIMENG.DLL - WINDOWS/APPPATCH/ACGENRAL.DLL - WINDOWS/SYSTEM32/WINMM.DLL - WINDOWS/SYSTEM32/MSACM32.DLL - WINDOWS/SYSTEM32/VERSION.DLL - WINDOWS/SYSTEM32/SHELL32.DLL - WINDOWS/SYSTEM32/SHLWAPI.DLL - WINDOWS/SYSTEM32/USERENV.DLL - WINDOWS/SYSTEM32/UXTHEME.DLL - WINDOWS/SYSTEM32/IMM32.DLL - …

…

And the list keeps on going for many more lines. It has been shortened for brevity.

5.4.7 Windows system restores

The following is a sample of a final timeline output for various Windows system restores:

exploited.dd--> 2010 02 25 Thu 16:46:14 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1315 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 02 26 Fri 01:00:19 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1316 created - Software Distribution Service 3.0 (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 02 27 Sat 01:08:44 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1317 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 02 28 Sun 02:08:44 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1318 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 03 01 Mon 03:08:43 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1319 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 03 02 Tue 12:00:27 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1320 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 03 03 Wed 15:18:02 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1321 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 03 04 Thu 16:20:39 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1322 created - System Checkpoint (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

exploited.dd--> 2010 03 05 Fri 09:45:30 | 0 | macb | 0 | 0 | 0 | 10143 | "[Restore Point] (Created) Restore point RP1324 created - Installed Windows 7 USB/DVD Download Tool (file: /media/ntfs//System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903})"

5.4.8 Windows shortcuts

The following is a sample of a final timeline output from various Windows shortcuts:

exploited.dd--> 2007 01 22 Mon 13:13:07 | 832 | m... | 0 | 0 | 0 | 87802 | "[Shortcut LNK] (Modified/Access/Created) C:/Documents and Settings/weiner/My Documents/Winter2001/George.xls <-/media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106096.LNK- which is stored on a local vol type - Fixed-SN 0x94492d7f - Rel path: ../../../../My Documents/Winter2001/George.xls [a rel. path str-SI ID exists-points to a file or dir] - mod since last backup (file: /media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106096.LNK)"

exploited.dd--> 2007 01 22 Mon 13:13:08 | 832 | .a.. | 0 | 0 | 0 | 87802 | "[Shortcut LNK] (Modified/Access/Created) C:/Documents and Settings/weiner/My Documents/Winter2001/George.xls <-/media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106096.LNK- which is stored on a local vol type - Fixed-SN 0x94492d7f - Rel path: ../../../../My Documents/Winter2001/George.xls [a rel. path str-SI ID exists-points to a file or dir] - mod since last backup (file: /media/ntfs/System

Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106096.LNK)"

exploited.dd--> 2007 01 22 Mon 13:15:21 | 680 | m... | 0 | 0 | 0 | 88309 | "[Shortcut LNK] (Modified/Access/Created) C:/Documents and Settings/weiner/My Documents/Winter2001/Winter2001.xls <-/media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106350.lnk- which is stored on a local vol type - Fixed- SN 0x94492d7f - Rel path: ../My Documents/Winter2001/Winter2001.xls Working dir: C:/Documents and Settings/weiner/My Documents/Winter2001 [a rel. path str-SI ID exists-working dir.-points to a file or dir] - mod since last backup (file: /media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106350.lnk)"

exploited.dd--> 2007 01 22 Mon 13:37:42 | 680 | .a.. | 0 | 0 | 0 | 88309 | "[Shortcut LNK] (Modified/Access/Created) C:/Documents and Settings/weiner/My Documents/Winter2001/Winter2001.xls <-/media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106350.lnk- which is stored on a local vol type - Fixed- SN 0x94492d7f - Rel path: ../My Documents/Winter2001/Winter2001.xls Working dir: C:/Documents and Settings/weiner/My Documents/Winter2001 [a rel. path str-SI ID exists-working dir.-points to a file or dir] - mod since last backup (file: /media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106350.lnk)"

exploited.dd--> 2007 01 22 Mon 13:47:10 | 680 | ma.. | 0 | 0 | 0 | 20965 | "[Shortcut LNK] (Modified/Access/Created) C:/Documents and Settings/weiner/My Documents/Winter2001/Winter2001.xls <-/media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106352.lnk- which is stored on a local vol type - Fixed- SN 0x94492d7f - Rel path: ../My Documents/Winter2001/Winter2001.xls Working dir: C:/Documents and Settings/weiner/My Documents/Winter2001 [a rel. path str-SI ID exists-working dir.-points to a file or dir] - mod since last backup (file: /media/ntfs/System Volume Information/_restore{0A1D308C-5063-4365-9E40-27C41DA5F903}/RP1374/A0106352.lnk)"

5.4.9 Windows Internet Explorer history files

The following is a sample of a final timeline output for various Windows Internet Explorer history files:

exploited.dd--> 2005 05 10 Tue 08:59:26 | 0 | m... | 0 | 0 | 0 | 737620 | "[Internet Explorer] (Content viewed/Content saved to drive) URL:http://img.dell.com/images/global/statichome/di_A4C9E7FA.jpg cache stored in: 6TCFAPSX/di_A4C9E7FA[1].jpg - HTTP/1.1 200 OK - Content-Length: 18542 - Content-Type: image/jpeg - ETag: ""cd2f2a68f54c51:8e3a5"" - X-Powered-By: ASP.NET (file: /media/ntfs/Weiner/Documents and Settings/rSskie/Local Settings/Temporary Internet Files/Content.IE5/index.dat)"

exploited.dd--> 2005 05 10 Tue 08:59:26 | 0 | m... | 0 | 0 | 0 | 737620 | "[Internet Explorer] (Content viewed/Content saved to drive)

URL:http://img.dell.com/images/global/statichome/di_DC25DC0B.jpg cache stored in: IJOLA5U7/di_DC25DC0B[1].jpg - HTTP/1.1 200 OK - Content-Length: 17750 - Content-Type: image/jpeg - ETag: ""216b2568f54c51:8e3a5"" - X-Powered-By: ASP.NET (file: /media/ntfs/Weiner/Documents and Settings/rSskie/Local Settings/Temporary Internet Files/Content.IE5/index.dat)"

exploited.dd--> 2005 05 10 Tue 08:59:27 | 0 | m... | 0 | 0 | 0 | 737620 | "[Internet Explorer] (Content viewed/Content saved to drive) URL:http://img.dell.com/images/global/brand/icons/viewlarger.gif cache stored in: ATUNA1IJ/viewlarger[1].gif - HTTP/1.1 200 OK - Content-Length: 187 - Content-Type: image/gif - ETag: ""050fc79456bc31:9c642"" - X-Powered-By: ASP.NET (file: /media/ntfs/Weiner/Documents and Settings/rSskie/Local Settings/Temporary Internet Files/Content.IE5/index.dat)"

exploited.dd--> 2005 05 10 Tue 08:59:27 | 0 | m... | 0 | 0 | 0 | 737620 | "[Internet Explorer] (Content viewed/Content saved to drive) URL:http://img.dell.com/images/global/brand/ui/arrow_top.gif cache stored in: 2XCDIHAD/arrow_top[1].gif - HTTP/1.1 200 OK - Content-Length: 48 - Content-Type: image/gif - ETag: ""0fd578a8d4bc31:8e3a5"" - X-Powered-By: ASP.NET (file: /media/ntfs/Weiner/Documents and Settings/rSskie/Local Settings/Temporary Internet Files/Content.IE5/index.dat)"

exploited.dd--> 2005 05 10 Tue 08:59:27 | 0 | m... | 0 | 0 | 0 | 737620 | "[Internet Explorer] (Content viewed/Content saved to drive) URL:http://img.dell.com/images/global/masthead/secondary_sep.gif cache stored in: IJOLA5U7/secondary_sep[1].gif - HTTP/1.1 200 OK - Content-Length: 78 - Content-Type: image/gif - ETag: ""80eed7208d4bc31:9c642"" - X-Powered-By: ASP.NET (file: /media/ntfs/Weiner/Documents and Settings/rSskie/Local Settings/Temporary Internet Files/Content.IE5/index.dat)"

exploited.dd--> 2005 05 10 Tue 08:59:27 | 0 | m... | 0 | 0 | 0 | 737620 | "[Internet Explorer] (Content viewed/Content saved to drive) URL:http://img.dell.com/images/global/statichome/di_3F01463E.jpg cache stored in: 2XCDIHAD/di_3F01463E[1].jpg - HTTP/1.1 200 OK - Content-Length: 21669 - Content-Type: image/jpeg - ETag: ""27e03a68f54c51:9c66e"" - X-Powered-By: ASP.NET (file: /media/ntfs/Weiner/Documents and Settings/rSskie/Local Settings/Temporary Internet Files/Content.IE5/index.dat)"

5.4.10 Firefox history files

The following is a sample of a final timeline output for various Firefox history files:

exploited.dd--> 2005 05 23 Mon 08:35:41 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark Folder [Dell] (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

exploited.dd--> 2005 05 23 Mon 08:35:42 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark URL Customize Links (http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&ar=CLinks) [redir.dll]

count 1 (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

exploited.dd--> 2005 05 23 Mon 08:35:42 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark URL Dell (http://www.dell.com/) [www.dell.com] count 0 (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

exploited.dd--> 2005 05 23 Mon 08:35:42 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark URL Free Hotmail (http://www.microsoft.com/isapi/redir.dll?prd=ie&ar=hotmail) [redir.dll] count 1 (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

exploited.dd--> 2005 05 23 Mon 08:35:42 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark URL MSN.com (http://www.microsoft.com/isapi/redir.dll?prd=ie&pver=6&ar=IStart) [redir.dll] count 0 (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

exploited.dd--> 2005 05 23 Mon 08:35:42 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark URL Radio Station Guide (http://www.microsoft.com/isapi/redir.dll?prd=windows&sbp=mediaplayer&plcid=&pver=6.1&os=&over=&olcid=&clcid=&ar=Media&sba=RadioBar&o1=&o2=&o3=) [redir.dll] count 0 (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

exploited.dd--> 2005 05 23 Mon 08:35:42 | 0 | ma.. | 0 | 0 | 0 | 760219 | "[Firefox 3 history] (dateAdded/LastModified) User: Weiner Bookmark URL Support.Dell.com (http://support.dell.com/) [support.dell.com] count 0 (file: /media/ntfs/Weiner/Documents and Settings/Weiner/Application Data/Mozilla/Firefox/Profiles/1b7fqd8y.default/places.sqlite)"

5.4.11 Windows Setupapi logs

The following is a sample of a final timeline output for various Windows Setupapi logs:

exploited.dd--> 2009 04 20 Mon 10:39:17 | 0 | macb | 0 | 0 | 0 | 3602 | "[SetupAPI Log] (Entry written) DriverContext: Reported hardware ID(s) from device parent bus. Context: Reported compatible identifiers from device parent bus. Context: Driver install entered (through services.exe). Information: Compatible INF file found. Information: Install section. Context: Processing a DIF_SELECTBESTCOMPATDRV request. Information: [c:/windows/inf/volume.inf]. Information: . Information: . Information: . Context: Processing a DIF_SELECTBESTCOMPATDRV request. Information: Copy-only installation [STORAGE/REMOVABLEMEDIA/7&2F5BE86A&0&RM]. Context: Processing a DIF_SELECTBESTCOMPATDRV request. Information: . Context: Processing a DIF_SELECTBESTCOMPATDRV request. Context: Installation in progress [c:/windows/inf/volume.inf]. Information: . Context: Processing a

DIF_SELECTBESTCOMPATDRV request. Information: [STORAGE/REMOVABLEMEDIA/7&2F5BE86A&0&RM]. Information: Device successfully setup [STORAGE/REMOVABLEMEDIA/7&2F5BE86A&0&RM]. (file: /media/ntfs//WINDOWS/setupapi.log)"

exploited.dd--> 2009 05 25 Mon 09:37:40 | 0 | macb | 0 | 0 | 0 | 3602 | "[SetupAPI Log] (Entry written) Context: Driver install entered (through services.exe). Error: . (file: /media/ntfs//WINDOWS/setupapi.log)"

exploited.dd--> 2009 05 25 Mon 09:39:39 | 0 | macb | 0 | 0 | 0 | 3602 | "[SetupAPI Log] (Entry written) DriverContext: Reported hardware ID(s) from device parent bus. Context: Reported compatible identifiers from device parent bus. Context: Driver install entered. Information: [c:/windows/inf/oem3.inf]. Information: . Information: . (file: /media/ntfs//WINDOWS/setupapi.log)"

exploited.dd--> 2009 05 25 Mon 09:39:40 | 0 | macb | 0 | 0 | 0 | 3602 | "[SetupAPI Log] (Entry written) DriverContext: Reported hardware ID(s) from device parent bus. Context: Reported compatible identifiers from device parent bus. Context: Driver install entered. Information: [c:/windows/inf/ich5core.inf]. Information: . Information: . (file: /media/ntfs//WINDOWS/setupapi.log)"

exploited.dd--> 2009 05 25 Mon 09:39:40 | 0 | macb | 0 | 0 | 0 | 3602 | "[SetupAPI Log] (Entry written) DriverContext: Reported hardware ID(s) from device parent bus. Context: Reported compatible identifiers from device parent bus. Context: Driver install entered. Information: [c:/windows/inf/iidigrcvdrvx2k.inf]. Information: . Information: . (file: /media/ntfs//WINDOWS/setupapi.log)"

5.4.12 Flash cookies

The following is a sample of a final timeline output for various Flash cookies:

exploited.dd--> 2008 06 24 Tue 12:42:42 | 0 | macb | 0 | 0 | 0 | 596173 | "[Flash Cookie] (timeStamp) timeStamp -> File: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/plugins/swf/preloader-F7.swf/2788-cookie.sol and object name: 2788-cookie variable: {timeStamp = (nan) } (file: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/plugins/swf/preloader-F7.swf/2788-cookie.sol)"

exploited.dd--> 2008 06 24 Tue 12:46:22 | 0 | macb | 0 | 0 | 0 | 598857 | "[Flash Cookie] (timeStamp) timeStamp -> File: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/plugins/swf/preloader-F7.swf/2790-cookie.sol and object name: 2790-cookie variable: {timeStamp = (nan) } (file: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/plugins/swf/preloader-F7.swf/2790-cookie.sol)"

exploited.dd--> 2008 08 12 Tue 18:28:31 | 0 | macb | 0 | 0 | 0 | 598480 | "[Flash Cookie] (timeStamp) timeStamp -> File: /media/ntfs//Documents and

Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/plugins/swf/preloader-F7.swf/2805-cookie.sol and object name: 2805-cookie variable: {timeStamp = (nan) } (file: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/plugins/swf/preloader-F7.swf/2805-cookie.sol)"

exploited.dd--> 2008 08 18 Mon 09:06:10 | 0 | macb | 0 | 0 | 0 | 105527 | "[Flash Cookie] (now) now -> File: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/includes/flash/favoritelocations/fav.swf/favLoc.sol and object name: favLoc variable: {now = (nan) } (file: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/52ZCX5EQ/www.gogole.com/includes/flash/favoritelocations/fav.swf/favLoc.sol)"

exploited.dd--> 2008 10 01 Wed 23:32:40 | 0 | macb | 0 | 0 | 0 | 736113 | "[Flash Cookie] (expires) expires -> File: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/8BD63CD4/msn.net/Kia_100108.sol and object name: Kia_100108 variable: {expires = (nan) } (file: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/8BD63CD4/msnpremads.edgesuite.net/Kia_100108.sol)"

exploited.dd--> 2008 10 18 Sat 08:58:02 | 0 | macb | 0 | 0 | 0 | 499160 | "[Flash Cookie] (expires) expires -> File: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash Player/#SharedObjects/8BD63CD4/.msn.com/NBC_Crusoe_101.sol and object name: NBC_Crusoe_101 variable: {expires = (nan) } (file: /media/ntfs//Documents and Settings/Weiner/Application Data/Macromedia/Flash

6 Conclusion

Digital forensic timelines can and should be used as a part of the investigative process whenever conducting a digital forensic investigation. However, as seen in this technical memorandum, their generation and use is not straightforward as there are many tools and platforms to choose from. This technical memorandum, with its included scripts and background material, has hopefully provided sufficient resources so that the reader can generate his own script or program based on his specific requirements.

Open source software was chosen for generating and processing timelines due to the fact that this increases the software and tools at the investigator's disposal which he can then use to customize not only the look and feel of the timeline but perform pattern matching and analysis against it. For these reasons, the authors assert that when combining *log2timeline* with The Sleuth Kit and simple shell scripts, an ideal manner software suite is available for the advanced generation of digital timelines. Timeline generation capability can be further customized by using additional data processing tools which could be simple C programs or shell scripts to make up for deficiencies in *log2timeline* or The Sleuth Kit.

Although graphical timeline generation is possible, the vast majority of those examined in this technical memorandum are severely lacking in capability. Most of the timeline generation tools examined are not particularly adept at generating or visualizing timelines. Although *log2timeline* is the most proficient tool for text-based timeline generation, it has no visualization capability whatsoever. The tool with the most potential for bridging the gap between timeline generation and visualization is *Aftertime* tool, which supports nearly as many date/time metadata sources as *log2timeline*. However, *Aftertime* has important limitations of its own including the inability to correctly export graphical or text-based reports for external data processing. In all fairness however, it is by and far the best timeline visualization software currently available, even though it is entirely incompatible with the Bodyfile format.

As such, much work has yet to be completed in the arena of timeline generation and analysis. Progress could be made sooner if software developers and forensic professionals could agree on an open standard for storing raw digital filesystem object date/time metadata, perhaps using the Bodyfile format as an option since it is already both open source in nature and an unofficial open standard due to the wide use of The Sleuth Kit which is gaining acceptance. Agreement on an open standard would provide additional incentive for the more popular digital forensic suites including EnCase and FTK to better support timeline generation and visualization within their own software products. Standardization would allow these suites to import and export to and from an agreed upon open standard such that additional data processing could be conducted using other software without the need for timeline conversion or translation.

Looking to the future, it would be advantageous to investigators if a definitive list of the effects of filesystem activity could be compiled, similar to those found in Annex A.3, but more complete and which extends across multiple filesystem formats. Moreover, filesystem developers could further help digital forensic professionals by incorporating additional date/time metadata structures into their existing filesystems, which could then be used for improving the resolution and collection of filesystem date/time metadata activity.

The work presented herein, including not only the technical background but also the various scripts and C programs, has been especially prepared for those working with Windows-based filesystems, but can be readily adapted for use with Linux and UNIX disk images without significant effort.

This page intentionally left blank.

References

[1] Guidance Software. Guidance Software Encase Forensic Edition. Informational web site. Digitalintelligence.com. http://www.digitialintelligence.com/software/guidancesoftware/encase/.

[2] AccessData. AccessData Forensic Toolkit. Informational web site. AccessData. 2010 http://accessdata.com/products/forensic-investigation/ftk.

[3] Dahon, Indra. Live Forensics: Forensic Tool: EnCase or FTK. Informational web site. Liveforensic.blogspot.com. September 2009. http://liveforensic.blogspot.com/2009/09/forensic-tool-encase-or-ftk.html.

[4] X-Ways Software Technology AG. X-Ways Forensics: Integrated Computer Forensics Software. Informational web site. X-Ways Software Technology AG. http://www.x-ways.net/forensics.

[5] Carrier, Brian. The Sleuth Kit: Description. Informational web site. Sleuthkit.org. 2010. http://www.sleuthkit.org/sleuthkit/desc.php.

[6] Wikipedia. ISO 9660. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ISO_9660.

[7] Wikipedia. Universal Disk Format. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Universal_Disk_Format.

[8] Wang, Wenguang. Wenguang's Introduction to Universal Disk Format (UDF). Informational web site. Wenguang Weng. February 2009. http://homepage.mac.com/wenguangwang/myhome/udf.html.

[9] Optical Storage Technology Association. Universal Disk Format Specification. Technical report. Optical Storage Technology Association. March 2005. http://www.osta.org/specs/pdf/udf260.pdf.

[10] Farmer, Dan, and Venema, Wietse. Forensic Discovery. Book. First edition. Addison-Wesley Publishing. January 2005. ISBN-13: 978-0201634976.

[11] Farmer, Dan. What Are Mactimes? Online article. Dr. Dobb's Journal. October 2000. http://drdobbs.com/184404275.

[12] Weise, Joel and Powell, Brad. Using Computer Forensics When Investigating System Attacks. Sun Blueprint/Technical report. Revision 1.0. Part No.: 819-2262-10. Sun Microsystems. April 2005. http://www.sun.com/blueprints/0405/819-2262.pdf.

[13] O'Keefe, Patrick B. Installing The Coroner's Toolkit and using the mactime utility. Technical document. University of South Carolina, Department of Computer Science. http://www.csc.sc.edu/~okeefe/tutorials/cert/i046.01.html.

[14] Wikipedia. Comparison of file systems. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Comparison_of_file_systems.

[15] Carrier, Brian. bodyfile. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Body_file.

[16] Unknown author. I see what you did there: Time stamps in digital forensics. Presentation. Unknown date. http://trustedsignal.com/presos/forensic_time_lines.pdf.

[17] Carrier, Brian. Mactime output. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Mactime_output.

[18] Wikipedia. MAC times. Online encyclopaedic entry. Wikimedia Foundation Inc. November 2010. http://en.wikipedia.org/wiki/MAC_times.

[19] Wikipedia. File Allocation Table. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/File_Allocation_Table.

[20] Wikipedia. Ext3. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Ext3.

[21] Wikipedia. NTFS. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/NTFS.

[22] Watson, Bob. Accdate. Informational web site. Bob Watson. June 2000. http://www.lagmonster.org/docs/DOS7/x-accdate.html.

[23] Microsoft Corporation. Microsoft Extensible Firmware Initiative FAT32 File System Specification: FAT: General Overview of On-Disk Format. Version 1.03. Whitepaper. Microsoft Corporation. December 2000.

[24] Newbigin, John. John's spec of the second extended filesystem. Informational web site. John Newbigin. http://uranus.chrysocome.net/explore2fs/es2fs.htm.

[25] Ionescu, Alex. NTFS On-Disk Structure: VisualBasic NTFS Programmer's Guide. Technical guide. Relsoft Technologies. 2004. http://www.alex-ionescu.com/NTFS.pdf.

[26] Wikipedia. Inode. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Inode.

[27] Wikipedia. Inode pointer structure. Online encyclopaedic entry. Wikimedia Foundation Inc. July 2010. http://en.wikipedia.org/wiki/Inode_pointer_structure.

[28] Carrier, Brian. File System Forensic Analysis. Book. First edition. Addison Wesley Publishing. 2005. ISBN: 0-32-126817-2.

[29] Pate, Steve D. UNIX Filesystems: Evolution, Design, and Implementation (Veritas Series). Book. First edition Wiley Publishing. 2003. ISBN: 0-471-16483-6.

[30] Wikipedia. Stat (Unix). Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/stat_(Unix).

[31] Plaugher, P.J. The Standard C Library. Book. First edition. 1991. ISBN.:0-13-131509-9.

[32] Wikipedia. ReiserFS. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ReiserFS.

[33] Buchholz, Florian. The structure of the Reiser file system. Technical document. Florian Buchholz. January 2006. http://homes.cerias.purdue.edu/~florian/reiser/reiserfs.php.

[34] Wikipedia. Unix time. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Unix_time.

[35] Wikipedia. Extent (file systems). Online encyclopaedic entry. Wikimedia Foundation Inc. November 2010. http://en.wikipedia.org/wiki/Extent_(file_systems).

[36] Wikipedia. Ext2. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Ext2.

[37] Wikipedia. Ext4. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Ext4.

[38] Wikipedia. Unix File System. Online encyclopaedic entry. Wikimedia Foundation Inc. November 2010. http://en.wikipedia.org/wiki/Unix_File_System.

[39] Wikipedia. Filesystem in Userspace. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Filesystem_in_Userspace.

[40] Quale, Doug, Lu, H.J, et al. Mount man file. Man page. Util-linux-ng. Version 0.97.3.

[41] Hewlett-Packard. HP-UX Release 11.0 System Calls and File Formats Sections 2 and 4. Technical reference. Edition 1. Volume 3 of 5. Document No.: B2355-90166. 1997. http://docs.hp.com/en/B2355-90682.pdf.

[42] Wikipedia. Fork (filesystem). Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Fork_(filesystem).

[43] Carbone, Richard. File recovery and data extraction using automated data recovery tools: A balanced approach using Windows and Linux when working with an unknown disk image and filesystem. Technical note. Defence R&D Canada - Valcartier. TN 2009-161. September 2009.

[44] Carbone, Richard. Developing a comprehensive approach for conducting a computer forensic investigation under Linux: A generic approach for maximum evidentiary extraction in a broad scope investigation (Draft). Technical Memorandum (Draft). Defence R&D Canada - Valcartier.

[45] Compaq Information Technologies Group. Guide to OpenVMS File Applications. Technical guide. Compaq Information Technologies Group. 2002. Order Number: AA-PV6PE-TK. http://h71000.www7.hp.com/doc/731final/documentation/pdf/ovms_731_file_app.pdf.

[46] Compaq Information Technologies Group. OpenVMS Guide to Extended File Specifications. Technical guide. Compaq Information Technologies Group. 2002. Order Number: AA-REZRB-TE. http://h71000.www7.hp.com/doc/731final/documentation/pdf/ovms_731_efs_gd.pdf.

[47] Wikipedia. Filesystem permissions. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/File_system_permissions.

[48] Wikipedia. Attrib. Online encyclopaedic entry. Wikimedia Foundation Inc. August 2010. http://en.wikipedia.org/wiki/Attrib.

[49] Wikipedia. Cacls. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Cacls.

[50] Wikipedia. Access control list. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Access_control_list.

[51] Klein, Helge. Command Line-Version (SetACL.exe) Syntax and Description. Technical description/informational web site. Helgeklein.com. 2011. http://helgeklein.com/setacl/documentation/command-line-version-setacl-exe/.

[52] ECMA International. Volume and File Structure of CDROM for Information Interchange. Technical report. Second edition. ECMA International. December 1987. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-119.pdf.

[53] Custer, Helen. Inside the Windows NT File System. Book. First edition. Microsoft Press. 1994. ISBN: 1-55615-660-X.

[54] Wikipedia. Standard C library. Online encyclopaedic entry. Wikimedia Foundation Inc. February 2011. http://en.wikipedia.org/wiki/Standard_C_library.

[55] Wikipedia. POSIX C library. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/C_POSIX_library.

[56] Wikipedia. POSIX. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/C_POSIX_library.

[57] Wikipedia. CD-RW. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/wiki/CD-RW.

[58] Wikipedia. DVD+RW. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/DVD%2BRW.

[59] Bennett, Hugh. Understanding Recordable & Rewritable DVD. Technical guide. First edition. April 2004. Optical Storage Technology Association. http://www.osta.org/technology/pdf/dvdqa.pdf.

[60] Shullich, Robert. Reverse Engineering the Microsoft Extended FAT File System (exFAT). Technical report. SANS Institute. December 2009. http://www.sans.org/reading_room/whitepapers/forensics/reverse-engineering-microsoft-exfat-file-system_33274.

[61] Wikipedia. exFAT. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ExFAT.

[62] Wikipedia. Extended file attributes. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/wiki/Extended_file_attributes.

[63] André, Jean-Pierre. NTFS-3G: Extended Attributes. Informational web site. Tuxera.com. January 2011. http://b.andre.pagesperso-orange.fr/extended-attr.html.

[64] Eager, Bob. Implementation of extended attributes on the FAT file system. Informational web site. Tavi Systems. October 2000. http://www.tavi.co.uk/os2pages/eadata.html.

[65] NTFS.com. NTFS File System Structure, Recovery Software, Hard Disk Internals. Informational web site. NTFS.com. 2011. http://www.ntfs.com.

[66] Gudjonsson, Kristinn. Log2timeline. Informational web site. Log2timeline.net. 2010. http://log2timeline.net/.

[67] Cloppert, Michael. Building a complete timeline for intrusion cases. Blog. SANS Institute. December 2008. http://computer-forensics.sans.org/blog/2008/12/29/building-a-complete-timeline-for-intrusion-cases/.

[68] Gudjonsson, Kristinn. Mastering the Super Timeline with log2timeline. Technical report. SANS Institute. June 2010. http://www.sans.org/reading_room/whitepapers/logging/mastering-super-timeline-log2timeline_33438.

[69] Gudjonsson, Kristinn. Mastering the Super Timeline: log2timeline style. Presentation. The 2010 European Community Digital Forensics and Incident Response Summit, London 2010. 2010. http://computer-forensics.sans.org/summit-archives/2010/files/eu-digital-forensics-incident-response-summit-kristinn-gudjonsson-mastering-the-super-timeline.pdf.

[70] Howlett, Tony. Open Source Security Tools. Book. First edition. Prentice Hall. July 2004. No.: 0-321-19443-8.

[71] Dowling, Anthony. Digital Forensics: A Demonstration of the Effectiveness of The Sleuth Kit and Autopsy Forensic Browser. Master's thesis. University of Otago, Dunedin, New Zealand. May 2006. http://eprints.otago.ac.nz/357/1/DowlingAcombOcr.pdf.

[72] Dowling, Anthony. The Sleuth Kit v2.01 and Autopsy Forensic Browser Demonstration. Technical guide. June 2006.

[73] Grundy, Barry J. The Law Enforcement and Forensic Examiner's Introduction to Linux: A Practitioner's Guide to Linux as a Computer Forensic Platform. Book. Version 3.78. December 2008. http://www.linuxleo.com/Docs/linuxintro-LEFE-3.78.pdf.

[74] Ford, Michael T. Analyses of Italian Malware, Romanian Rootkits, and United States Computer Law. Technical report. SANS Institute. March 2003.

[75] Miller, III, Roland E. Analysis of an unknown Mac OS X Public Beta System Using Mac OS X 10. Technical report. SANS Institute. September 2002.

[76] Jeffris, Clarke L. The Coroners Toolkit – In depth. Technical report. SANS Institute. 2002.

[77] Naval Postgraduate School. Fiwalk. Informational web site. DEEP: Digital Evaluation and Exploitation, Department of Computer Science, Naval Postgraduate School. http://domex.nps.edu/deep/Fiwalk.html.

[78] Wikipedia. Compact Disc. Online encyclopaedic entry. Wikimedia Foundation Inc. April 2011. http://en.wikipedia.org/wiki/Compact_Disc.

[79] Carvey, Harlan. TimeLine Analysis, part III. Blog. Windowsir.blogspot.com. February 2009. http://windowsir.blogspot.com/2009/02/timeline-analysis-pt-iii.html.

[80] Carvey, Harlan. Timeline Analysis, part VI - Taking Another Step. Blog. Windowsir.blogspot.com. April 2009. http://windowsir.blogspot.com/2009/04/timeline-analysis-pt-vi-taking-another.html.

[81] Mauro, Jim and McDougall, Richard. Solaris Internals: Core Kernel Components. Book. First edition. Sun Microsystems Press. 2000. ISBN: 0-13-022496-0.

[82] Wikipedia. Exchangeable image file format. Online encyclopaedic entry. Wikimedia Foundation Inc. June 2011. http://en.wikipedia.org/wiki/Exchangeable_image_file_format.

[83] ArcSight Inc. Common Event Format. Technical Note. Revision 15. July 2009. ArcSight Inc.

[84] ArcSight Inc. Common Event Format: Event Interoperability Standard. White paper. 2006. ArcSight Inc.

Annex A About CDs, disc images formats and filesystem-based MAC times

A.1 Windows CD-ROM optical installation media vs. detected filesystem type and size

The following is a table listing the various Windows optical installation media with their approximate respective sizes. This list is not complete but suitably conveys the point that many DVDs actually contain ISO 9660-based filesystems.

Table 2. Corresponding Windows optical installation media detected filesystem versus approximate optical disc size.

Windows operating system installation optical disc media	Detected filesystem	Approximate size
Windows NT Server (no SP)	ISO 9660	< 650 MB
Windows 2000 (no SP)	ISO 9660	< 650 MB
Windows XP 32-bit Professional SP2	ISO 9660	< 650 MB
Windows XP 64-bit Professional SP1	ISO 9660	< 650 MB
Windows 2003 Server Enterprise SP1	ISO 9660	< 650 MB
Windows Vista 32-bit Ultimate SP1	ISO 9660	> 3.0 GB
Windows Vista 64-bit Ultimate SP1	ISO 9660	> 3.8 GB
Windows 7 32-bit Ultimate (no SP)	ISO 9660	< 2.0 GB
Windows 7 64-bit Ultimate (no SP)	ISO 9660	> 3.0 GB
Windows 2008 64-bit Server (no SP)	ISO 9660	> 2.9 GB

A.2 Disc image format for various publicly available Linux, BSD and Solaris distributions

According to [78], the maximum data capacity of a standard Compact Disc is 870 MB or 912,261,120 bytes. Based on a variety of collected distributions by the authors over the years including Linux, BSD and Solaris, only disc images larger than 870 MB have been analyzed. The analysis carried out herein is the same as that carried in the previous section – verify the size and disc image format of the disc image in question. The results can be found in the table below:

Table 3. Disc image-based distribution with detected filesystem, operating system type and disc image size.

Stored disc image name	Detected filesystem	Operating system type	Size (in bytes)
samurai-0.9.5.iso	ISO 9660	Linux (Samurai)	1,410,062,336

Icaros-pc-i386.iso	ISO 9660	AROS Research OS/AmigaOS	1,980,651,520
backtrack4-R2.iso	ISO 9660	Linux (BackTrack 4)	2,034,880,512
sol-10u9-ga-x86-dvd.iso	ISO 9660	Solaris (x86) 10 10/09	2,146,959,360
FreeBSD-8.1-RELEASE-amd64-dvd1.iso	ISO 9660	FreeBSD	2,305,976,320
FC-4-x86_64-DVD.iso	ISO 9660	Linux (Fedora Core 4)	2,932,650,688
FC-14-x86_64-DVD.iso	ISO 9660	Linux (Fedora Core 14)	3,520,802,816
Solaris_9_SPARC_1202.iso	ISO 9660	Solaris (SPARC) 9 12/02	3,633,971,200
Knoppix_V6.2DVD-2009-11-18-EN.iso	ISO 9660	Linux (Knoppix 6.2)	3,853,285,376
Knoppix511DVDEnglish.iso	ISO 9660	Linux (Knoppix 5.11)	4,324,202,496
Snow_Leopard_Mac_OSX.iso	ISO 9660	Mac OS X 10.6	7,771,521,024

Thus, this simple analysis reveals that many of today's popular UNIX-related software distributions which although written to DVD due to their size are in fact based on the CDFS filesystem format.

A.3 MAC times for various filesystems given different actions taken against files

It is difficult to correctly deduce both the exact meaning of *mtime*, *atime*, *ctime* and *crtime* and understand how different filesystem actions affect these date/time metadata. Sources of reliable information are scarce and sometimes conflicting. Therefore, according to [17], the following chart applies to the various date/time metadata for the most commonly found filesystems handled by The Sleuth Kit:

Table 4. Mac Meaning by File System (reproduced from [17]).

File System	m (*mtime*)	a (*atime*)	c (*ctime*)	b (*crtime*)
Ext2/3	Modified	Accessed	Changed	Does not apply
FAT	Written	Accessed	Does not apply	Created
NTFS	File Modified	Accessed	MFT Modified	Created
UFS	Modified	Accessed	Changed	Does not apply

According to [16, 28], the NTFS filesystem will update both its $STANDARD_INFORMATION and $FILE_NAME MFT entries differently, depending on the underlying action. The following charts clearly demonstrate the date/time consequences of various actions undertaken either by the user or the operating system.

Table 5. File modifications resulting in changes to the MFT date/time $STANDARD_INFORMATION attribute (reproduced from [16]).

	Rename	Local Move	Volume Move	Copy	Access	Modify	Create	Delete
M						X	X	
A			X	X	X	X	X	
C	X	X	X	X			X	X
B				X			X	

Table 6. File modifications resulting in changes to the MFT date/time $FILE_NAME attribute (reproduced from [16]).

	Rename	Local Move	Volume Move	Copy	Access	Modify	Create	Delete
M		X	X	X			X	X
A			X	X			X	
C		X	X	X			X	X
B			X	X			X	

Thus, based on the aforementioned charts, concluding that the NTFS filesystem is complex cannot be overstated as it is arguably the most complex filesystem the forensic investigator is likely to encounter. Moreover, in comparison to inode-based filesystems, there are many potentially extractable metadata attributes which can be of service to the investigator depending on the specific needs of a given investigation.

Table 7. File deletion MAC time changes for a given filesystem type (Source [28]).

	FAT	NTFS[54]	EXT2/3/4	UFS
M			X	X
A				
C			X	X
B				
D[55]			X	

54 Reference [28] makes no mention of whether these changes are reflected in $STANDARD_INFORMATION or $FILE_NAME. $STANDARD_INFORMATION is assumed.

55 D denotes *dtime*.

Table 8. Directory-based MAC time changes upon sub-file deletion for a given filesystem type (Source [28]).

	FAT	NTFS[56]	EXT2/3/4	UFS
M		X	X	X
A		X	X	X
C		X	X	X
B				
D[57]				

The information provided by the aforementioned charts is highly revealing although unfortunately certain discrepancies become apparent. These discrepancies are concerned specifically with the MAC time changes that occur to an NTFS-based file upon its deletion. This discrepancy can be seen by examining the NTFS-related MAC times of tables 3 and 5.

Further definitive research is required in order to decisively determine the MAC time changes which occur to files given different actions taken against them. Although it is important that the NTFS filesystem be better understood with respect to which types of changes results in which date/time metadata modifications, other filesystems including Ext2/3/4, XFS and UFS should be included in such research, given their widespread use across many diverse UNIX-like systems.

The work of Brian Carrier from [28] will be considered, if only for the intents and purposes herein, as correct.

56 Reference [28] makes no mention of whether these changes are reflected in $STANDARD_INFORMATION or $FILE_NAME. $STANDARD_INFORMATION is assumed.

57 D denotes *dtime*.

Annex B Shell scripts and C code

B.1 Shell script: *timeline.sh*

The following is an example of the Bash shell script used to extract timeline-related metadata from a raw hard disk drive image, *evidence.dd*. Line numbers have been added to improve readability. Specific C programs written by the authors, which are highlighted for easy location in the script, can be found in their corresponding sections further on in this Annex.

Use of this code is pursuant to the terms specified in the Disclaimer (see Page viii) and C source code and Bash shell script code disclosure licensing agreement (see Page xi).

```
1   #!/bin/sh
2
3   #Clear screen:
4   #=============
5   clear
6
7   #Print disclaimer:
8   #=================
9   echo "Welcome to TIMELINE.SH, a Log2timeline/TSK-based filesystem timeline
    automated generation system."
10  echo "Written by Richard Carbone, DRDC Valcartier, 2010-2011."
11  sleep 2
12
13  #Test user input:
14  #================
15
16  if [ "$1" == "-h" ] || [ "$1" == "-help" ] || [ "$1" == "--help" ]
17  then
18      echo ""
19      echo "======================================================"
20      echo "To use the program correctly, specify the following:"
21      echo "======================================================"
22      echo ""
23      echo   "./timeline.sh    Working_Directory    Byte_Offset    Disk_Image_File
    Mount_Point Partition#"
24      echo ""
25      echo "Where:"
26      echo "======"
27      echo ""
28      echo "Working_Directory = Specify absolute directory location where timeline
    will be built."
29      echo "-> Location must actually exist."
30      echo ""
31      echo "Byte_Offset = Byte offset where filesystem begins."
```

```
32      echo "-> Bytes to start of actual filesystem."
33      echo ""
34      echo "Disk_Image_File = Absolute location and name of evidence disk image
    to mount."
35      echo "-> i.e. /tmp/evidence.dd"
36      echo ""
37      echo "Mount_Point = Absolute location and name of evidence disk image
    mount point."
38      echo "-> i.e. /media/evidence"
39      echo ""
40      echo "Partition# = current evidence disk image partition number."
41      echo "-> i.e. 0"
42      exit 1
43  fi
44
45  echo ""
46  echo "Testing if Working Directory location exists..."
47  if [ -d "$1" ]
48      then
49          echo "Working Directory OK."
50      else
51          echo "Working Directory does not exist."
52          echo "Exiting..."
53          exit 2
54  fi
55
56  echo ""
57  echo "Testing if Filesystem Byte Offset is equal to 0 or greater..."
58  if [ "$2" -ge "0" ]
59      then
60          echo "Filesystem Byte Offset OK."
61      else
62          echo "Could not determine status of Filesystem Byte Offset."
63          echo "Exiting..."
64          exit 3
65  fi
66
67  echo ""
68  echo "Testing if Disk Image File exists..."
69  if [ -f "$3" ]
70      then
71          echo "Disk Image File OK."
72      else
73          echo "Could not determine the status of the Disk Image File."
74          echo "Exiting..."
75          exit 4
76  fi
77
```

```
 78  echo ""
 79  echo "Testing if Disk Image File is empty..."
 80  if [ `ls -al "$3" | awk '{print $5}'` -gt 0 ]
 81      then
 82          echo "Disk Image File size OK."
 83      else
 84          echo "Disk Image File is empty."
 85          echo "Exiting..."
 86          exit 5
 87  fi
 88
 89  echo ""
 90  echo "Testing if Mount Point location exists..."
 91  if [ -d "$4" ]
 92      then
 93          echo "Mount Point location OK."
 94      else
 95          echo "Mount Point location does not exist."
 96          echo "Exiting..."
 97          exit 6
 98  fi
 99
100  echo ""
101  echo "Testing if current Partition Number is equal to 0 or greater..."
102  if [ "$5" -ge "0" ]
103      then
104          echo "Partition Number OK."
105      else
106          echo "Could not determine Partition Number."
107          echo "Exiting..."
108          exit 7
109  fi
110
111  echo  ""
112  echo "Starting Program "$0"..."
113
114  #Assigning input parameters to easy-of-use variables:
115  #====================================================
116  location=$1
117  offset=$2
118  disk_image_name=$3
119  image_name=`/bin/basename $3`
120  mount_point=$4
121  part_number=$5
122  fls_offset=$(($offset/512))
123
124  #Mount filesystem image:
125  #=======================
```

```
126 echo ""
127 echo "Mounting disk image read-only..."
128 mount -o ro,loop,utf8,uni_xlate=1,offset=$offset $disk_image_name
    $mount_point
129
130 #Make Directory Structure:
131 #========================
132 echo ""
133 echo "Creating timeline directory structure..."
134 cd $location
135 mkdir timeline
136 mkdir timeline/$image_name
137 mkdir timeline/$image_name/$part_number
138 mkdir timeline/$image_name/$part_number/mac
139 mkdir timeline/$image_name/$part_number/time
140 mkdir timeline/$image_name/$part_number/final_timeline
141 mkdir timeline/$image_name/$part_number/regfiles
142 mkdir timeline/$image_name/$part_number/eventlogs
143 mkdir timeline/$image_name/$part_number/eventxlogs
144
145 #Generate MAC Times:
146 #===================
147 echo ""
148 echo "Generating MAC Times from The Sleuth Kit..."
149 cd $location/timeline/$image_name/$part_number
150 fls -m / -r -a -l -p -o $fls_offset $disk_image_name > mac/all_files.mac
151 fls -m / -r -d -l -p -o $fls_offset $disk_image_name > mac/deleted_files.mac
152 fls -m / -r -u -l -p -o $fls_offset $disk_image_name > mac/undeleted_files.mac
153
154 #Convert MAC times to Time Body:
155 #===============================
156 echo ""
157 echo "Converting MAC times to Body Format..."
158 cd $location/timeline/$image_name/$part_number/mac
159 mactime -b all_files.mac   -d -y -m -z EST5EDT | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / |
    sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / >
    ../time/all_files.time
160 mactime -b deleted_files.mac   -d -y -m -z EST5EDT | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / > ../time/deleted_files.time
161 mactime -b undeleted_files.mac -d -y -m -z EST5EDT | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ /  > ../time/undeleted_files.time
162
163 #Remove all MAC duplicates:
164 #Some MAC objects may in double between all_files and deleted_files
165 #=========================================================
166 echo ""
```

```
167 echo "Removing all duplicates..."
168 cd $location/timeline/$image_name/$part_number/time
169 cat *.time | sort | uniq > files_timeline_sort_uniq.time
170
171 #Find all pertinent registry files:
172 #================================
173 echo ""
174 echo "Finding and processing registry files..."
175 cd $location/timeline/$image_name/$part_number
176 find $mount_point -depth -print0 | xargs -0 file | grep "MS\ Windows\ registry\ file"
    | grep -v "NtServicePackUninstall" | grep -v "Uninstall" > registry.txt
177 file_name_type_line_parser registry.txt reg
178 cat reg | awk '{print "find \""$0"\"  -depth -print | cpio  -pamVd \"regfiles"$0"\""}' >
    sh.sh
179 sh sh.sh
180 rm sh.sh registry.txt
181 mv reg regfiles_found.txt
182 find regfiles/ -type f -print | awk '{print "perl /usr/local/bin/regtime.pl -r \""$0"\" >>
    mac/regtime.mac"}' > regtime.sh
183 sh regtime.sh
184 cd mac/
185 mactime -b regtime.mac -d -y -m -z EST5EDT | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / |
    sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    ../time/regtime.time
186 cd ..
187 rm regtime.sh
188
189 #Find all pertinent EVT Event Log files:
190 #=======================================
191 echo ""
192 echo "Finding and processing EVT event logs..."
193 cd $location/timeline/$image_name/$part_number
194 find $mount_point/ -depth -type f -print > out
195 find_eventlog_signature out > out2
196 cat out2 | grep "Windows Event Log" | grep -v "signature verification program." >
    events
197 file_name_type_line_parser events events2
198 cat events2 | awk '{print "find \""$0"\"  -depth -print | cpio -pamVd
    \"eventlogs"$0"\""}' > sh.sh
199 sh sh.sh
200 rm sh.sh out2 events out
201 mv events2 eventlogs_found.txt
202 find eventlogs/ -type f -print | awk '{print "log2timeline -z EST5EDT -f evt \""$0"\" -
    o mactime -w mac/eventlogs.mac"}' > eventlogs.sh
203 sh eventlogs.sh
204 rm eventlogs.sh
205 cd mac
```

```
206 cat eventlogs.mac | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \
    / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sort > out
207 unixtime_to_systime
208 rm out
209 mv eventlogs.time ../time
210
211 #Find all pertinent EVTX Event Log files:
212 #=======================================
213 echo ""
214 echo "Finding and processing EVTX event logs..."
215 cd $location/timeline/$image_name/$part_number
216 find $mount_point -depth -print0 | xargs -0 file | grep "MS\ Windows\ Vista\ Event\
    Log" | grep -v "NtServicePackUninstall" | grep -v "Uninstall" | grep -v ", empty" >
    eventxlogs.txt
217 file_name_type_line_parser eventxlogs.txt eventx
218 cat eventx | awk '{print "find \""$0"\" -depth -print | cpio -pamVd
    \"eventxlogs"$0"\""}' > sh.sh
219 sh sh.sh
220 rm sh.sh eventxlogs.txt
221 mv eventx eventxlogs_found.txt
222 find eventxlogs/ -type f -print | awk '{print "log2timeline -z EST5EDT -f evtx
    \""$0"\" -o mactime -w mac/eventxlogs.mac"}' > eventxlogs.sh
223 sh eventxlogs.sh
224 cd mac/
225 mactime -b eventxlogs.mac -d -y -m -z EST5EDT | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /
    | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /
    > ../time/eventxlogs.time
226 cd ..
227 rm eventxlogs.sh
228
229 #Prefetch:
230 #=========
231 echo ""
232 echo "Processing Windows prefetch..."
233 log2timeline -z EST5EDT -f prefetch $mount_point/WINDOWS/Prefetch/ -o
    mactime -w mac/prefetch.mac
234 mactime -b mac/prefetch.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/prefetch.time
235
236 #System Restore:
237 #===============
238 echo ""
239 echo "Processing Windows restore point..."
240 find $mount_point/System\ Volume\ Information/ -type d | grep "\{" | grep "\}" |
    grep -v snapshot | grep -v "\RP" | sort | uniq > out
241 cat out | awk '{print "log2timeline -z EST5EDT -f restore \""$0"\" -o mactime -w
    mac/restore.mac"}' > sh.sh
```

```
242 sh sh.sh
243 mactime -b mac/restore.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/restore.time
244 rm out sh.sh
245
246 #Windows Shortcuts:
247 #==================
248 echo ""
249 echo "Processing shortcuts..."
250 find $mount_point -type f -print0 | xargs -0 file | grep -i -P '(MS\ Windows\
    shortcut)' > shortcuts
251 file_name_type_line_parser shortcuts shortcuts2
252 cat shortcuts2 | awk '{print "log2timeline -z EST5EDT -f win_link \""$0"\" -o
    mactime -w mac/shortcuts.mac"}' > sh.sh
253 sh sh.sh
254 rm shortcuts sh.sh
255 mv shortcuts2 shortcuts_found.txt
256 mactime -b mac/shortcuts.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\
    \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/shortcuts.time
257
258 #Windows IE History.DAT
259 #======================
260 echo ""
261 echo "Processing Internet Explorer history..."
262 find $mount_point -type f -print0 | xargs -0 file | grep -i -P '(Internet\ Explorer\
    cache\ file)' > ieindex
263 file_name_type_line_parser ieindex ieindex2
264 cat ieindex2 | awk '{print "log2timeline -z EST5EDT -f iehistory \""$0"\" -o mactime
    -w mac/iehistory.mac"}' > sh.sh
265 sh sh.sh
266 mv ieindex2 ieindex_found.txt
267 rm ieindex sh.sh
268 mactime -b mac/iehistory.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\
    \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/iehistory.time
269
270 #Search for Firefox history files
271 #================================
272 echo ""
273 echo "Processing Firefox history..."
274 find $mount_point -type f -iname "places.sqlite" > firefox3
275 cat firefox3 | awk '{print "log2timeline -z EST5EDT -f firefox3 \""$0"\" -o mactime -
    w mac/firefox3.mac"}' > sh.sh
276 sh sh.sh
277 mv firefox3 firefox3_found.txt
278 rm sh.sh
```

```
279 mactime -b mac/firefox3.mac -d -y -m |sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/firefox3.time
280
281 #Search for Setupapi Logs
282 #=======================
283 echo ""
284 echo "Processing setupapi logs..."
285 find $mount_point/ -type f -print | grep -i -P '(setupapi\.log|setupapi\.del)' >
    setupapi
286 cat setupapi | awk '{print "log2timeline -z EST5EDT -f setupapi \""$0"\" -o
    mactime -w mac/setupapi.mac"}' > sh.sh
287 sh sh.sh
288 mv setupapi setupapi_found.txt
289 rm sh.sh
290 mactime -b mac/setupapi.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\
    \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/setupapi.time
291
292 #Search for Windows firewall logs
293 #================================
294 echo ""
295 echo "Processing firewall logs..."
296 find $mount_point/ -type f -print | grep -i -P '(pfirewall\.log.)' > firewall
297 cat firewall | awk '{print "log2timeline -z EST5EDT -f xpfirewall \""$0"\" -o mactime
    -w mac/firewall.mac"}' > sh.sh
298 sh sh.sh
299 mv firewall firewall_found.txt
300 rm sh.sh
301 mactime -b mac/firewall.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \
    \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/firewall.time
302
303 #Search for Flash cookies
304 #=======================
305 echo ""
306 echo "Processing Flash cookies..."
307 find $mount_point/ -type f -iname *.sol -print0 | xargs -0 file | grep data | grep -v
    "ASCII English text" > flash_cookies
308 file_name_type_line_parser flash_cookies flash_cookies2
309 cat flash_cookies2 | awk '{print "log2timeline  -z EST5EDT -f sol \""$0"\" -o
    mactime -w mac/flash.mac"}' > sh.sh
310 sh sh.sh
311 mv flash_cookies2 flash_cookies_found.txt
312 rm flash_cookies sh.sh
313 mactime -b mac/flash.mac -d -y -m | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\
    \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ / | sed s/,/\ \ \|\ \ /  >
    time/flash.time
```

```
314
315 #Generate final timeline listing:
316 #================================
317 echo ""
318 echo "Generating final timeline..."
319 cd time
320 cat setupapi.time firewall.time flash.time firefox.time eventlogs.time
    files_timeline_sort_uniq.time regtime.time iehistory.time shortcuts.time
    restore.time prefetch.time | sort | uniq | grep -v "Date Size Type Mode UID GID
    Meta File Name" > ../final_timeline/$image_name.$part_number.time.temp
321
322 file_name=../final_timeline/$image_name.$part_number.time.temp
323
324 while read line
325 do
326   echo "$image_name --> $line" >>
    ../final_timeline/$image_name.$part_number.time
327 done <$file_name
328 rm ../final_timeline/$image_name.$part_number.time.temp
329
330 #Unmount filesystem:
331 #===================
332 echo ""
333 echo "Unmounting "$mount_point" = "$disk_image_name", offset = "$offset
334 umount -d $mount_point
```

B.2 C programs source code

B.2.1 C program: *file_name_type_line_parser.c*

Use of this code is pursuant to the terms specified in the Disclaimer (see Page viii) and C source code and Bash shell script code disclosure licensing agreement (see Page xi).

```
1   /* This program aims to remove the file type description of a filetypes
2    * file generated by the UNIX 'file' command.  The file type portion of the
3    * 'file' program's output is denoted by the ":" character.  This program
4    * finds that character and removes the rest of the line.
5    *
6    * However, to speed things up a bit this program will also perform some
7    * character substitution and replace all "$" characters it encounters with
8    * the escaped version "\$".
9    */
10  /* This program was written by Richard Carbone, DRDC Valcartier - 2010.
     * Last updated August 2011.
     * All rights reserved - Her Majesty the Queen in Right of Canada, as represented by
     * the Minister of National Defence, 2011.
     *
     * THIS PROGRAM CLEANS UP FIND/FILE COMMAND OUTPUT
     */
11  #include <stdio.h>
12  #include <stdlib.h>
13  #include <string.h>
14
15  FILE *file_types;
16  FILE *file_rpt;
17
18  int main(int argc, char **argv)
19  {
20    int argnum = 0;
21    int raw_byte = 0;
22    int bytes_read = 0;
23    char line_buffer[1024];
24
25    fprintf(stderr, "\nProgram written by Richard Carbone.\n");
26    fprintf(stderr, "DRDC Valcartier, 2011.\n\n");
27
28    if (argc != 3)
29    {
30      fprintf(stderr, "Error, program usage: %s [inputfilename] [outputfilename]\n",
    argv[0]);
31      exit(1);
32    }
```

```
33        if ((file_types = fopen(argv[1], "rb"))==NULL)
34        {
35          fprintf(stderr, "Error, unable to open input file %s\n", argv[1]);
36          exit(2);
37        }
38        file_rpt = fopen(argv[2], "wb");
39
40        while ((raw_byte = fgetc(file_types))!=EOF)
41        {
42          if (raw_byte != ':')
43          {
44            line_buffer[bytes_read] = raw_byte;
45            if (raw_byte == 36)
46            {
47              fprintf(file_rpt, "\\");
48              fprintf(file_rpt, "%c", line_buffer[bytes_read]);
49              bytes_read++;
50            }
51            else
52            {
53              fprintf(file_rpt, "%c", line_buffer[bytes_read]);
54              bytes_read++;
55            }
56          }
57          else if (raw_byte == ':')
58          {
59            memset(line_buffer, 0, sizeof(int) * 1024);
60            bytes_read = 0;
61            while ((raw_byte = fgetc(file_types))!='\n')
62            {
63            }
64            fprintf(file_rpt, "\n");
65          }
66        }
67
68        fclose(file_types);
69        fclose(file_rpt);
70
71        exit(0);
72      }
```

B.2.2 C program: *find_eventlog_signature.c*

Use of this code is pursuant to the terms specified in the Disclaimer (see Page viii) and C source code and Bash shell script code disclosure licensing agreement (see Page xi).

```
1   /* This program requires that find command be used to generate a file listing.
2      This program then reads in that file listing from the stdin and processes
3      its listed files to verify if they match the included file signature herein
4      below.  This program can deal with Windows filenames of all sorts.  Once
5      preliminary tests are done to verify that stdin infile is not EOF or NULL
6      processing begins.  The file is read byte by byte until a \n chararcter is
7      found which denotes a new filename from the infile.  Each byte is read as
8      an integer and is then copied to a buffer and upon reaching a \n the \n
9      character is not included in the buffer but is followed by a \0.  The filename
10     in the buffer is opened for reading and str_size size bytes are read in and
11     compared to the signature.  If a match is found the program states as much
12     otherwise it closes the file and proceeds to the next file.  */
13  /* This program was written by Richard Carbone, DRDC Valcartier - 2010.
       Last updated August 2011.
       All rights reserved - Her Majesty the Queen in Right of Canada, as represented by
       the Minister of National Defence, 2011.

       THIS PROGRAM DETECTS XP (AND PRIOR) EVENT LOGS
    */
14
15  #include <stdio.h>
16  #include <stdlib.h>
17  #include <string.h>
18
19  #define str_size 8
20  #define sig_size 9
21  #define buffer_size 1024
22
23  FILE *input_file;
24  FILE *read_sig_file;
25
26  int main(int argc, char **argv)
27  {
28     int raw_byte = 0;
29     int bytes_read = 0;
30     char line_buffer[buffer_size];
31     char buffer[buffer_size];
32     char sig_event_log[] = {'\x30','\x00','\x00','\x00','\x4C','\x66','\x4C','\x65','\0'};
33
34     fprintf(stdout, "\nWindows Event Log signature verification program.\n");
35     fprintf(stdout, "\nWritten by Richard Carbone.\nDRDC Valcartier.\n2010.\n");
36     fprintf(stdout, "\n\n");
```

```
37
38          if (argc != 2)
39          {
40            fprintf(stderr, "Error, program usage: %s [inputfilename]\n", argv[0]);
41            exit(1);
42          }
43
44          if ((input_file = fopen(argv[1], "r")) == NULL)
45          {
46            fprintf(stderr, "Error, unable to open input file %s\n", argv[1]);
47            exit(1);
48          }
49
50          while ((raw_byte = fgetc(input_file)) != EOF) //read from inputfile until EOF or \n
51          {  if (raw_byte != '\n')                    //once \n reached copy to line_buffer
52            {  line_buffer[bytes_read] = raw_byte;     //and then open that file for reading
53              bytes_read++;
54            }
55            else if (raw_byte == '\n')
56            {  line_buffer[bytes_read] = '\0';
57
58              read_sig_file = fopen(line_buffer, "rb");
59
60              if (NULL != fgets(buffer, sig_size, read_sig_file))
61              {
62                //fprintf(stderr, "File empty\n");
63                //memset(line_buffer, 0, sizeof(char) * buffer_size);
64                //break;
65              }
66
67              if (memcmp(buffer, sig_event_log, str_size) == 0)
68              {  fprintf(stdout, "%s: Windows Event Log\n", line_buffer);
69              }
70
71              fclose(read_sig_file);
72              memset(line_buffer, 0, sizeof(char) * buffer_size);
73              bytes_read = 0;
74            }
75          }
76          fclose(input_file);
77
78          return(0);
79        }
```

B.2.3 C program: *unixtime_to_systime.c*

Use of this code is pursuant to the terms specified in the Disclaimer (see Page viii) and C source code and Bash shell script code disclosure licensing agreement (see Page xi).

```
0       /* This program was written by Richard Carbone, DRDC Valcartier - 2010.
         * Last updated August 2011.
         * All rights reserved - Her Majesty the Queen in Right of Canada, as represented by
         * the Minister of National Defence, 2011.
         *
         * THIS PROGRAM CONVERTS UNIX TIME TO SYSTEM TIME
         */
1       #include <time.h>
2       #include <stdio.h>
3       #include <stdlib.h>
4
5       #define SIZE 256
6       #define BUFF_SIZE 4096
7
8       FILE *evtlog;
9       FILE *output;
10
11      int main (void)
12      {
13        char buffer[SIZE];
14        char message[BUFF_SIZE];
15        int raw_byte = 0;
16        char *tzone;
17        time_t timestamp;
18        char* format;
19        double filetime;
20        char buff1[10];
21        char buff2[100];
22        char buff3[100];
23
24        evtlog = fopen("out", "r");
25        output = fopen("eventlogs.time", "w");
26        while (fscanf(evtlog, "%lf", &filetime) != EOF)
27        {
28          timestamp = filetime;
29
30          format = "%Y %m %d %a %T";
31
32          tzone="TZ=EST";
33          putenv(tzone);
34          strftime(buffer, SIZE, format, localtime(&timestamp));
35          fprintf(output, "%s", buffer);
36          fscanf(evtlog, "%s", &buff1);
```

```
37            fscanf(evtlog, "%s", &buff2);
38            fscanf(evtlog, "%s", &buff3);
39            fprintf(output, "  %s  SID:%s\t\t\t\t\t\tMessage:", buff1, buff3);
40            while ((raw_byte = fgetc(evtlog)) != '\n')
41            {
42               fputc(raw_byte, output);
43            }
44            fprintf(output, "\n");
45         }
46
47         fclose(evtlog);
48         fclose(output);
49
50         return(0);
51      }
```

This page intentionally left blank.

Bibliography

AccessData. AccessData Forensic Toolkit. Informational web site. AccessData. 2010 http://accessdata.com/products/forensic-investigation/ftk.

André, Jean-Pierre. NTFS-3G: Extended Attributes. Informational web site. Tuxera.com. January 2011. http://b.andre.pagesperso-orange.fr/extended-attr.html.

Barika, Mridul Sankar, Guptab, Gaurav, et al. An efficient technique for enhancing forensic capabilities of Ext2 file system. Technical report. Elsevier Digital Investigations 4S (2007) S55-S61. 2007. http://www.dfrws.org/2007/proceedings/p55-barik.pdf.

Bennett, Hugh. Understanding Recordable & Rewritable DVD. Technical guide. First edition. April 2004. Optical Storage Technology Association. http://www.osta.org/technology/pdf/dvdqa.pdf.

Blunden, Bill and Below Gotham Labs. Anti-Forensics: The Rootkit Connection. Technical report. Black Hat USA 2009. 2009. http://www.blackhat.com/presentations/bh-usa-09/BLUNDEN/BHUSA09-Blunden-AntiForensics-PAPER.pdf.

Buchholz, Florian and Spafford, Eugene. On the role of file system metadata in digital forensics. Technical report. Journal of Digital Investigation, Vol. 1(4), pp. 297-308. December 2004.

Buchholz, Florian. The structure of the Reiser file system. Technical document. Florian Buchholz. January 2006. http://homes.cerias.purdue.edu/~florian/reiser/reiserfs.php.

Bunting, Steve, and Wei, William. EnCase Computer Forensics: The Official EnCE: EnCase Certified Examiner Study Guide. Book. First edition. Wiley Publishing. 2006. ISBN-13: 978-07821-4435-2.

Carbone, Richard. Developing a comprehensive approach for conducting a computer forensic investigation under Linux: A generic approach for maximum evidentiary extraction in a broad scope investigation (Draft). Technical Memorandum (Draft). Defence R&D Canada - Valcartier.

Carbone, Richard. File recovery and data extraction using automated data recovery tools: A balanced approach using Windows and Linux when working with an unknown disk image and filesystem. Technical note. Defence R&D Canada - Valcartier. TN 2009-161. September 2009.

Carrier, Brian. bodyfile. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Body_file.

Carrier, Brian. FAT. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://www.forensicswiki.org/wiki/FAT.

Carrier, Brian. File System Forensic Analysis. Book. First edition. Addison Wesley Publishing. 2005. ISBN: 0-32-126817-2.

Carrier, Brian. Fls. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Fls.

Carrier, Brian. Mactime output. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Mactime_output.

Carrier, Brian. Mactime. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Mactime.

Carrier, Brian. Metadata Address. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Metadata_address.

Carrier, Brian. NTFS Implementation Notes. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=NTFS_Implementation_Notes.

Carrier, Brian. NTFS. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://www.forensicswiki.org/wiki/NTFS.

Carrier, Brian. The Sleuth Kit Overview and Automated Scanning Features. Presentation. Sleuth Kit and Open Source Digital Forensics Conference June 9, 2010. June 2010. http://www.basistech.com/conference/2010/osdf-slides/carrier-sleuthkitoverview.pdf.

Carrier, Brian. The Sleuth Kit: Description. Informational web site. Sleuthkit.org. 2010. http://www.sleuthkit.org/sleuthkit/desc.php.

Carrier, Brian. Timelines. Informative/technical web page. Wiki.sleuthkit.org. January 2011. http://wiki.sleuthkit.org/index.php?title=Timeline.

Carvey, Harlan. Timeline Analysis Part 3 : Log2timeline. Blog. Windowsir.blogspot.com. March 2010. http://thedigitalstandard.blogspot.com/2010/03/timeline-analysis-part-3-log2timeline.html.

Carvey, Harlan. Timeline Analysis Part I : Creating a Timeline of a Live Windows System. Blog. Windowsir.blogspot.com. March 2010. http://thedigitalstandard.blogspot.com/2010/03/creating-timeline-of-live-windows.html.

Carvey, Harlan. TimeLine Analysis part II (Sources). Blog. Windowsir.blogspot.com. February 2009. http://windowsir.blogspot.com/2009/02/timeline-analysis-part-ii-sources.html.

Carvey, Harlan. TimeLine Analysis, part III. Blog. Windowsir.blogspot.com. February 2009. http://windowsir.blogspot.com/2009/02/timeline-analysis-pt-iii.html.

Carvey, Harlan. Timeline Analysis, part VI - Taking Another Step. Blog. Windowsir.blogspot.com. April 2009. http://windowsir.blogspot.com/2009/04/timeline-analysis-pt-vi-taking-another.html.

Carvey, Harlan. Timeline Creation and Analysis. Blog. Windowsir.blogspot.com. March 2010. http://windowsir.blogspot.com/2010/03/timeline-creation-and-analysis.html.

Carvey, Harlan. Windows Forensic Analysis DVD Toolkit. Book. First edition. Syngress Publishing. 2007. ISBN-13: 978-1-59749-156-3.

Casey, Eoghan. Handbook of Computer Crime Investigation: Forensic Tools and Technology. Book. First edition. Academic Press. 2003. ISBN: 0-12-163103-6.

Charter, Brandon. EVTX and Windows Event Logging. Technical report. SANS Institute. 2008. http://www.sans.org/reading_room/whitepapers/logging/evtx-windows-event-logging_32949.

Cloppert, Michael. Building a complete timeline for intrusion cases. Blog. SANS Institute. December 2008. http://computer-forensics.sans.org/blog/2008/12/29/building-a-complete-timeline-for-intrusion-cases/.

Cloppert, Michael. Ex-Tip: An Extensible Timeline Analysis Framework in Perl. Technical report. SANS Institute. May 2008. http://www.sans.org/reading_room/whitepapers/forensics/ex-tip-extensible-timeline-analysis-framework-perl_32767.

Compaq Information Technologies Group. Guide to OpenVMS File Applications. Technical guide. Compaq Information Technologies Group. 2002. Order Number: AA-PV6PE-TK. http://h71000.www7.hp.com/doc/731final/documentation/pdf/ovms_731_file_app.pdf.

Compaq Information Technologies Group. OpenVMS Guide to Extended File Specifications. Technical guide. Compaq Information Technologies Group. 2002. Order Number: AA-REZRB-TE. http://h71000.www7.hp.com/doc/731final/documentation/pdf/ovms_731_efs_gd.pdf.

Craiger, Philip and Burke, Paul K. Mac Forensics: Mac OS X and the HFS+ File System. Technical report. University of Central Florida and National Centre for Forensic Science. http://www2.tech.purdue.edu/cit/Courses/cit556/readings/MacForensicsCraiger.pdf.

Custer, Helen. Inside the Windows NT File System. Book. First edition. Microsoft Press. 1994. ISBN: 1-55615-660-X.

Dahon, Indra. Live Forensics: Forensic Tool: EnCase or FTK. Informational web site. Liveforensic.blogspot.com. September 2009. http://liveforensic.blogspot.com/2009/09/forensic-tool-encase-or-ftk.html.

Dowling, Anthony. Digital Forensics: A Demonstration of the Effectiveness of The Sleuth Kit and Autopsy Forensic Browser. Master's thesis. University of Otago, Dunedin, New Zealand. May 2006. http://eprints.otago.ac.nz/357/1/DowlingAcombOcr.pdf.

Dowling, Anthony. The Sleuth Kit v2.01 and Autopsy Forensic Browser Demonstration. Technical guide. June 2006.

Eager, Bob. Implementation of extended attributes on the FAT file system. Informational web site. Tavi Systems. October 2000. http://www.tavi.co.uk/os2pages/eadata.html.

EC-Council. Computer Hacking Forensic Investigator: Courseware Manual version 3.0. Courseware. Volume 1, 2, and 3. Third edition. EC-Council.

Eckstein, Knut. Forensics for Advanced UNIX File Systems. Technical report. NATO C3 Agency. Published in 2004 IEEE/USMA IA Workshop. 2004. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1437842.

ECMA International. Volume and File Structure of CDROM for Information Interchange. Technical report. Second edition. ECMA International. December 1987. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-119.pdf.

Fairbanks, Kevin D., Lee, Christopher P. and Henry L. Owen III. Forensic Implications of Ext4. Technical report. School of Electrical and Computer Engineering, Georgia Institute of Technology. 2010. http://delivery.acm.org/10.1145/1860000/1852691/a22-fairbanks.pdf?key1=1852691&key2=3418706921&coll=DL&dl=ACM&CFID=6873414&CFTOKEN=12508791.

Farmer, Dan, and Venema, Wietse. Forensic Discovery. Book. First edition. Addison-Wesley Publishing. January 2005. ISBN-13: 978-0201634976.

Farmer, Dan. What Are Mactimes? Online article. Dr. Dobb's Journal. October 2000. http://drdobbs.com/184404275.

Ford, Michael T. Analyses of Italian Malware, Romanian Rootkits, and United States Computer Law. Technical report. SANS Institute. March 2003.

Free 60. FATX. Online information resource. Free 60. December 2010. http://www.free60.org/FATX.

Gelinas, Jacques. UMSDOS HOW-TO. Howto guide. Version 1.2. The Linux Documentation Project. December 2001. ftp://ftp.eenet.ee/doc/LDP/HOWTO/UMSDOS-HOWTO.html.

Giampaolo, Dominic. Practical File System Design with the Be File System. Book. First edition. Morgan Kaufmann Publishers. 1999. ISBN: 1-55860-497-9.

Grundy, Barry J. The Law Enforcement and Forensic Examiner's Introduction to Linux: A Practitioner's Guide to Linux as a Computer Forensic Platform. Book. Version 3.78. December 2008. http://www.linuxleo.com/Docs/linuxintro-LEFE-3.78.pdf.

Gudjonsson, Kristinn. Artifact Timeline Creation and Analysis - part 2. Blog. SANS Institute. August 2009. http://computer-forensics.sans.org/blog/2009/08/14/artifact-timeline-creation-and-analysis-part-2/.

Gudjonsson, Kristinn. Log2timeline. Informational web site. Log2timeline.net. 2010. http://log2timeline.net/.

Gudjonsson, Kristinn. Mastering the Super Timeline with log2timeline. Technical report. SANS Institute. June 2010. http://www.sans.org/reading_room/whitepapers/logging/mastering-super-timeline-log2timeline_33438.

Gudjonsson, Kristinn. Mastering the Super Timeline: log2timeline style. Presentation. The 2010 European Community Digital Forensics and Incident Response Summit, London 2010. 2010.

http://computer-forensics.sans.org/summit-archives/2010/files/eu-digital-forensics-incident-response-summit-kristinn-gudjonsson-mastering-the-super-timeline.pdf.

Guidance Software. Guidance Software Encase Forensic Edition. Informational web site. Digitalintelligence.com. http://www.digitialintelligence.com/software/guidancesoftware/encase/.

Hewlett-Packard. HP-UX Release 11.0 System Calls and File Formats Sections 2 and 4. Technical reference. Edition 1. Volume 3 of 5. Document No.: B2355-90166. 1997. http://docs.hp.com/en/B2355-90682.pdf.

Hewlett-Packard. HP-UX Release 11.0 System Calls and File Formats Sections 2 and 4. Technical reference. Edition 1. Volume 3 of 5. Document No.: B2355-90166. 1997.

Hinner, Martin. Filesystems HOWTO. Howto guide. Version 0.8. The Linux Documentation Project. January 2007. http://www.ibiblio.org/pub/linux/docs/howto/other-formats/pdf/Filesystems-HOWTO.pdf.

Howlett, Tony. Open Source Security Tools. Book. First edition. Prentice Hall. July 2004. No.: 0-321-19443-8.

Ionescu, Alex. NTFS On-Disk Structure: VisualBasic NTFS Programmer's Guide. Technical guide. Relsoft Technologies. 2004. http://www.alex-ionescu.com/NTFS.pdf.

Jeffris, Clarke L. The Coroners Toolkit – In depth. Technical report. SANS Institute. 2002.

Klein, Helge. Command Line-Version (SetACL.exe) Syntax and Description. Technical description/informational web site. Helgeklein.com. 2011. http://helgeklein.com/setacl/documentation/command-line-version-setacl-exe/.

Kubasiak, Ryan R. Macintosh Forensics: A Guide for the Forensically Sound Examination of a Macintosh Computer. Technical guide. New York State Troopers. May 2007. http://www.appleexaminer.com/Downloads/MacForensics.pdf.

Lee, Rob. Digital Forensic SIFT'ing: Registry and Filesystem Timeline Creation. Blog. SANS Institute. February 2009. http://computer-forensics.sans.org/blog/2009/02/24/digital-forensic-sifting-registry-and-filesystem-timeline-creation/.

Lee, Rob. Shadow Timelines And Other VolumeShadowCopy Digital Forensics Techniques with the Sleuthkit on Windows. Blog. SANS Institute. March 2010. http://computer-forensics.sans.org/blog/2010/03/16/shadow-timelines-and-other-shadowvolumecopy-digital-forensics-techniques-with-the-sleuthkit-on-windows/?utm_source=rss&utm_medium=rss&utm_campaign=shadow-timelines-and-other-shadowvolumecopy-digital-forensics-techniques-with-the-sleuthkit-on-windows.

Lui, John C.S. Minix File System. Intro. Course notes (professor). http://www.edugrid.in/webfolder/OpSystems/9_FileSystems/Chinese_Univ/supplement_minix_fs.pdf.

Mauro, Jim and McDougall, Richard. Solaris Internals: Core Kernel Components. Book. First edition. Sun Microsystems Press. 2000. ISBN: 0-13-022496-0.

Microsoft Corporation. Microsoft Extensible Firmware Initiative FAT32 File System Specification: FAT: General Overview of On-Disk Format. Version 1.03. Whitepaper. Microsoft Corporation. December 2000.

Middleton, Bruce. Cyber Crime Field Handbook. Book. First edition. Auerbach Publishing. 2002. ISBN: 0-8493-1192-6.

Migletz, James. Automated Metadata Extraction. Master's thesis. Naval Postgraduate School. June 2008. http://simson.net/clips/students/08Jun_Migletz.pdf.

Miller, III, Roland E. Analysis of an unknown Mac OS X Public Beta System Using Mac OS X 10. Technical report. SANS Institute. September 2002.

Mueller, Lance. Detecting timestamp changing utilities. Blog. Www.forensickb.com. February 2009. http://www.forensickb.com/2009/02/detecting-timestamp-changing-utlities.html.

Naval Postgraduate School. Fiwalk. Informational web site. DEEP: Digital Evaluation and Exploitation, Department of Computer Science, Naval Postgraduate School. http://domex.nps.edu/deep/Fiwalk.html.

Nelson, Bill, Phillips, Amelia, and Christopher Steuart. Guide to Computer Forensics and Investigations. Book. Third edition. Cengage Learning. 2009. ISBN-13: 978-1-435-49883-9.

Newbigin, John. John's spec of the second extended filesystem. Informational web site. John Newbigin. http://uranus.chrysocome.net/explore2fs/es2fs.htm.

Nikkel, Bruce J. Forensic Analysis of GPT Disks and GUID Partition Tables. Technical report. The International Journal of Digital Forensics and Incident Response Vol. 6, No. 1-2 (doi:10.1016/j.diin.2009.07.001). November 2009.

NTFS.com. NTFS File System Structure, Recovery Software, Hard Disk Internals. Informational web site. NTFS.com. 2011. http://www.ntfs.com.

O'Keefe, Patrick B. Installing The Coroner's Toolkit and using the mactime utility. Technical document. University of South Carolina, Department of Computer Science. http://www.csc.sc.edu/~okeefe/tutorials/cert/i046.01.html.

Obialero, Roberto. Forensic Analysis of a Compromised Intranet Server. Technical report. SANS Institute. March 2006. http://www.sans.org/reading_room/whitepapers/forensics/forensic-analysis-compromised-intranet-server_1652.

Optical Storage Technology Association. Universal Disk Format Specification. Technical report. Optical Storage Technology Association. March 2005. http://www.osta.org/specs/pdf/udf260.pdf.

Pate, Steve D. UNIX Filesystems: Evolution, Design, and Implementation (Veritas Series). Book. First edition Wiley Publishing. 2003. ISBN: 0-471-16483-6.

Plaugher, P.J. The Standard C Library. Book. First edition. 1991. ISBN.:0-13-131509-9.

Poirier, Dave. The Second Extended File System: Internal Layout. Technical report. Dave Poirier. 2009. http://www.giis.co.in/ext2.pdf.

Quale, Doug, Lu, H.J, et al. Mount man file. Man page. Util-linux-ng. Version 0.97.3

Shinder, Debra Littlejohn and Tittel, Ed. Scene of the Cybercrime Computer Forensics Handbook. Book. Syngress Publishing. 2002. ISBN: 1-931836-65-5.

Shullich, Robert. Reverse Engineering the Microsoft Extended FAT File System (exFAT). Technical report. SANS Institute. December 2009. http://www.sans.org/reading_room/whitepapers/forensics/reverse-engineering-microsoft-exfat-file-system_33274.

Silicon Graphics Inc. IRIX System Administration I. Courseware. Silicon Graphics. May 2004. ISA1-1.4-SM.

Stephenson, Peter. Investigating Computer-Related Crime: A Handbook for Corporate Investigations. Book. First edition. CRC Press. 2000. ISBN: 0-8493-2218-9.

Summers, Clayton. Introduction to ISO 9660: what it is, how it is implemented, and how it has been extended. Technical report. Disc Manufacturing Inc. May 1995. http://www.pandreonline.com/documentos/cds/iso9660/norma.pdf.

Tweedie, Stephen. EXT3, Journalling Filesystem. Technical document. Stephen Tweedie. July 2000. http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html.

Unknown author. I see what you did there: Time stamps in digital forensics. Presentation. Unknown date. http://trustedsignal.com/presos/forensic_time_lines.pdf.

Venema, Wietse. Journaling file system forensics. Presentation. 2007.

Vogels, Werner. File system usage in Windows NT 4.0. Technical report. 17th ACM Symposium on Operating Systems Principles (SOSP'99), Published as Operating Systems Review 34(5):93-109, Dec. 1999. Department of Computer Science, Cornell University. December 1999.

Volonino, Linda and Anzaldua, Reynaldo. Computer Forensics for Dummies. Book. First edition. Wiley Publishing. 2008. ISBN: 978-0-470-37191-6.

Wang, Wenguang. Wenguang's Introduction to Universal Disk Format (UDF). Informational web site. Wenguang Weng. February 2009. http://homepage.mac.com/wenguangwang/myhome/udf.html.

Watson, Bob. Accdate. Informational web site. Bob Watson. June 2000. http://www.lagmonster.org/docs/DOS7/x-accdate.html.

Weise, Joel and Powell, Brad. Using Computer Forensics When Investigating System Attacks. Sun Blueprint/Technical report. Revision 1.0. Part No.: 819-2262-10. Sun Microsystems. April 2005. http://www.sun.com/blueprints/0405/819-2262.pdf.

Wikipedia. Access control list. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Access_control_list.

Wikipedia. Advanced Disc Filing System. Online encyclopaedic entry. Wikimedia Foundation Inc. July 2010. http://en.wikipedia.org/wiki/Advanced_Disc_Filing_System.

Wikipedia. Andrew Filing System. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Andrew_File_System.

Wikipedia. Attrib. Online encyclopaedic entry. Wikimedia Foundation Inc. August 2010. http://en.wikipedia.org/wiki/Attrib.

Wikipedia. Be File System. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Be_File_System.

Wikipedia. BeOS. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/BeOS.

Wikipedia. Btrfs. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Btrfs.

Wikipedia. Cacls. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Cacls.

Wikipedia. CD-RW. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/wiki/CD-RW.

Wikipedia. Comparison of file systems. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Comparison_of_file_systems.

Wikipedia. Cramfs. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Cramfs.

Wikipedia. DVD+RW. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/DVD%2BRW.

Wikipedia. Exchangeable image file format. Online encyclopaedic entry. Wikimedia Foundation Inc. June 2011. http://en.wikipedia.org/wiki/Exchangeable_image_file_format.

Wikipedia. exFAT. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ExFAT.

Wikipedia. Ext2. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Ext2.

Wikipedia. Ext3. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Ext3.

Wikipedia. Ext4. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Ext4.

Wikipedia. Extended file attributes. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/wiki/Extended_file_attributes.

Wikipedia. Extent (file systems). Online encyclopaedic entry. Wikimedia Foundation Inc. November 2010. http://en.wikipedia.org/wiki/Extent_(file_systems).

Wikipedia. File Allocation Table. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/File_Allocation_Table.

Wikipedia. Files-11. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Files-11.

Wikipedia. Filesystem in Userspace. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Filesystem_in_Userspace.

Wikipedia. Filesystem permissions. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/File_system_permissions.

Wikipedia. Fork (filesystem). Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Fork_(filesystem).

Wikipedia. HFS Plus. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/HFS_Plus.

Wikipedia. Hierarchical File. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Hierarchical_File_System.

Wikipedia. High Performance File System. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/High_Performance_File_System.

Wikipedia. Inode pointer structure. Online encyclopaedic entry. Wikimedia Foundation Inc. July 2010. http://en.wikipedia.org/wiki/Inode_pointer_structure.

Wikipedia. Inode. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Inode.

Wikipedia. Intrusion detection system. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/wiki/Intrusion_detection_system.

Wikipedia. ISO 9660. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ISO_9660.

Wikipedia. JFS (file system). Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/JFS_(file_system).

Wikipedia. Journaling file system. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Journaling_file_system.

Wikipedia. MAC times. Online encyclopaedic entry. Wikimedia Foundation Inc. November 2010. http://en.wikipedia.org/wiki/MAC_times.

Wikipedia. MINIX file system. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/MINIX_file_system.

Wikipedia. Network intrusion detection system. Online encyclopaedic entry. Wikimedia Foundation Inc. February 2011. http://en.wikipedia.org/wiki/Network_intrusion_detection_system.

Wikipedia. Next3. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Next3.

Wikipedia. NTFS. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/NTFS.

Wikipedia. POSIX C library. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/C_POSIX_library.

Wikipedia. POSIX. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2011. http://en.wikipedia.org/Posix.

Wikipedia. QNX4FS. Online encyclopaedic entry. Wikimedia Foundation Inc. March 2010. http://en.wikipedia.org/wiki/QNX4FS.

Wikipedia. ReiserFS. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ReiserFS.

Wikipedia. Romfs. Online encyclopaedic entry. Wikimedia Foundation Inc. June 2010. http://en.wikipedia.org/wiki/Romfs.

Wikipedia. Standard C library. Online encyclopaedic entry. Wikimedia Foundation Inc. February 2011. http://en.wikipedia.org/wiki/Standard_C_library.

Wikipedia. Stat (Unix). Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/stat_(Unix).

Wikipedia. Transaction-Safe FAT File System. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/TFAT.

Wikipedia. Universal Disk Format. Online encyclopaedic entry. Wikimedia Foundation Inc. December 2010. http://en.wikipedia.org/wiki/Universal_Disk_Format.

Wikipedia. Unix File System. Online encyclopaedic entry. Wikimedia Foundation Inc. November 2010. http://en.wikipedia.org/wiki/Unix_File_System.

Wikipedia. Unix time. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Unix_time.

Wikipedia. VERITAS File System. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/VERITAS_File_System.

Wikipedia. VHD (file format). Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/VHD_(file_format).

Wikipedia. Virtual disk image. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/Virtual_disk_image.

Wikipedia. VMDK. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/VMDK.

Wikipedia. ZFS. Online encyclopaedic entry. Wikimedia Foundation Inc. January 2011. http://en.wikipedia.org/wiki/ZFS.

X-Ways Software Technology AG. X-Ways Forensics: Integrated Computer Forensics Software. Informational web site. X-Ways Software Technology AG. http://www.x-ways.net/forensics.

List of symbols/abbreviations/acronyms/initialisms

ABI	Application Binary Interface
ADFS	Advanced Disc Filing System
ACL	Access Control List
ADS	Alternate Data Stream
AFF	Advanced Forensics Format
AFFS	Amiga Fast File System
AIX	Advanced Interactive eXecutive
AKA	Also Known As
API	Application Programming Interface
ANSI	American National Standards Institute
atime	Access time
AV	Anti-Virus
Bash	Bourne-again shell
BD-RW	Blu-Ray Disc – ReWritable
BeFS	Be File System
BeOS	Be Operating System
BSD	Berkeley Software Distribution
Btrfs	B-Tree Filesystem
CD	Compact Disc
CD-ROM	Compact Disc - Read Only Memory
CD-RW	Compact Disc – ReWritable
CDFS	Compact Disc File System / See ISO 9660
CEF	Common Exchange Format
CPU	Central Processing Unit
CramFS	Compressed ROM File System
crtime	Creation time
CSV	Comma-Separated Values
ctime	Last change time

DACL	Discretionary Access Control List
DEC	Digital Equipment Corporation
DNS	Domain Name Service
DOS	Disk Operating System
DVD	Digital Video Disc / Digital Versatile Disc
DVD-RW	Digital Video Disc – ReWritable / Digital Versatile Disc – ReWritable
EFS	Extent File System
EST5EDT	Eastern Standard Time, time zone 5, Eastern Daylight Time
EVT	Windows Event Log files (pre-Vista)
EVTX	Windows Event Log files (Vista and newer)
EWF	Export Witness Compression Format
EXIF	EXchangeable Image File Format
exFAT	Extended File Allocation Table
Ext2/3/4	Second Extended Filesystem/Third Extended Filesystem/Fourth Extended Filesystem
FAT12/FAT16/FAT32	File Allocation Table 12-bit/File Allocation Table 16-bit/File Allocation Table 32-bit
FFS	Fast File System
FTK	Forensic ToolKit
FUSE	Filesystem in Userspace
GB	Gigabyte
GID	Group ID
GUI	Graphical User Interface
HFS	Hierarchical File System
HFS+	Hierarchical File System Plus/+
HPFS	High Performance File System
HTML	HyperText Markup Language
I/O	Input/Output
IBM	International Business Machine Inc.
ID	Identification

IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
ISO 9660	International Organization for Standardization 9660
JFS	Journaled File System
MAC	Modify, Access, Change
MAC	Media Access Control (address)
MB	Megabyte
MFT	Master File Table
mtime	Modification time
NIDS	Network Intrusion Detection System
NTFS	New Technology File System
OpenVMS	Open Virtual Memory System
OS	Operating System
OS/2	Operating System/2
PC	Personal Computer
PCAP	Packet CAPture
Perl	Practical Extraction and Report Language
POSIX	Portable Operating System Interface for UNIX
QNX4FS	QNX System 4 File System
R&D	Research & Development
RAID	Redundant Array of Inexpensive Disks
Reiser4	Reiser 4
ReiserFS	Reiser File System
RFS	See ReiserFS
RISC	Reduced Instruction Set Code
ROM	Read-Only Memory
ROMFS	Read-Only Memory File System / ROM File System
SAM	Security Accounts Manager
SANS	SysAdmin, Audit, Networking, and Security
SDXC	Secure Digital eXtended Capacity

SGI	Silicon Graphics Inc.
SGID	Setgid/Set Group ID
SID	System ID (Windows)
SP/SP1	Service Pack/Service Pack 1
SUID	Setuid/Set User ID
SysV	System V
TB	Terabyte
TLN	TimeLiNe
TFAT	Transaction-safe FAT
UDF	Universal Disk Format
UFS	UNIX file system
UID	User ID
UTC	Universal Time, Coordinated
VAX	Virtual Address eXtension
VDI	Virtual Disk Image
VHD	Virtual Hard Disk
VMDK	Virtual Machine Disk
VxFS	Veritas File System
XFS	eXtended File System
XML	eXtensible Markup Language
ZFS	Zettabyte File System

Glossary

ACL (Access Control List)

An ACL is a list of permissions which can be associated to files, directories or other operating system-specific objects. ACLs are used to assign additional permissions to system objects in order to specify in a more fine-tuned manner the permissions associated to said object(s). ACLs are generally applied on top of DACL-based permissions.

ADFS (Advanced Disk Filing System)

ADFS is a filesystem native to Acorn and RISC OS computer systems. This filesystem is natively supported by Linux.

AFFS (Amiga Fast File System)

AFFS is a filesystem native to the Amiga personal computer. This filesystem is natively supported by Linux.

Bash (Bourne-again shell)

See Command Shell for more information.

BeOS (Be Operating System)

BeOS is an early 1990s PC-based operating system which was to compete against other popular operating systems at the time. Its native filesystem is BeFS.

BeFS (Be File System)

BeFS is a filesystem for the PC-based BeOS operating system. It was a highly advanced filesystem for its time. This filesystem is natively supported by Linux.

Bodyfile

The Bodyfile format is a preliminary timeline format which is used as a mid-point between a filesystem's raw date/time metadata structures and a readable text-based timeline. It was first proposed by Farmer/Venema and used with their forensic software suite TCT and is also used by TSK as its intermediate timeline format. It is a more complete and complex intermediate format than TLN.

Btrfs (B-Tree Filesystem)

Btrfs is an advanced native Linux filesystem developed by Chris Mason of Oracle Corp. which is set to eventually supersede the Ext4 filesystem. This filesystem is natively supported by Linux.

C

C is a high-level computer programming language and is the basis for all UNIX-based operating systems which rely extensively on its built-in library functions. Different standards

have existed over the years but C is now governed by ANSI and has subsequently become an ANSI standardized programming language.

CDFS (Compact Disc File System)

CDFS is the standard filesystem for CD-based optical media although it is easily extended to DVD-based media. This filesystem is natively supported by Linux.

Coherent

Coherent is an older UNIX System V filesystem found on some SysV systems.

Command Shell

The Command Shell is the system command line interpreter and can, under UNIX systems, take the form of many different command shells including *Bash*, *ksh*, *csh*, *tcsh*, *zsh*, *ash* and many others. The Command Shell interprets the commands input by the user or operator and translates them into actionable commands for the operating system kernel which is then requested to perform some useful work on behalf of the user or operator.

CramFS (Compressed ROM File System)

CramFS is a filesystem similar to ROMFS except that it is a compressed ROM filesystem and it is commonly found in embedded devices. This filesystem is natively supported by Linux.

DACL (Discretionary Access Control List)

A DACL is a series of permissions generally attributed to files and directories but can sometimes be applied to other operating system objects. DACLs define the permissions that users, groups and the public have with respect to various system objects. DACLs differ from ACLs in that they are the default permissions applied to objects whereas ACLs expand upon default object permissions by providing or removing additional authorizations.

DOS (Disk Operating System)

DOS is a very common computer operating system found in the 1980s and early 90s. It was the operating system of choice on PCs until 32-bit versions of Windows came to market. It is now outdated and has not been improved upon since the mid 1990s; however, it is still prevalent for those conducting troubleshooting and diagnostic-related work on a PC. Microsoft was the largest developer and distributor of DOS, although other alternatives existed.

DOS attributes

Four specific DOS attributes exist. The first is *R* for read-only. An attribute of *A* states the file is ready for archiving. Finally, *S* and *H* state that the file is a system or hidden file, respectively. DOS Attributes can be associated to files and directories.

EFS (Extent File System)

EFS is a native filesystem to older SGI workstations, servers and supercomputers. It is commonly found as the default filesystem on SGI software installation CD-ROMs. This filesystem is natively supported by Linux.

EnCase

EnCase is a computer digital forensic software developed and sold by Guidance Software which is very popular among law enforcement.

Epoch time

Epoch time, also called UNIX time, starts January 1, 1970, 12:00 am UTC. It is a second-base time system and is the accumulation of seconds since the beginning of Epoch time until a given moment in the present. It is also commonly known as UNIX or POSIX time and in its 32-bit form is set to expire in 2038.

EXIF (EXchangeable Image File Format)

A digital standard used for storing and extracting information tags from digital files including but not limited to images, videos, documents, etc.

exFAT (Extended File Allocation Table)

exFAT is a Windows supported 64-bit implementation of FAT. It is supported by Linux as a FUSE-based filesystem.

Ext2 (Second Extended Filesystem)

Ext2 is a native filesystem of the Linux operating system. This filesystem is natively supported by Linux.

Ext3 (Third Extended Filesystem)

Ext3 is a native filesystem based of the Linux operating system based on Ext2 which provides a filesystem journaling capability. This filesystem is natively supported by Linux.

Ext4 (Fourth Extended Filesystem)

Ext4 is the latest Ext-based filesystem of the Linux operating system which supersedes Ext2/3. It continues providing filesystem journaling. It also provides greater performance, reliability and allows for much larger file and filesystem sizes. This filesystem is natively supported by Linux.

Extent

An extent is a chunk or piece of filesystem space which is contiguous in nature and is specifically used for storage. Unlike an inode, an extent does not store information about a given file. Extents are used to further reduce filesystem fragmentation.

Extended attribute

Extended attributes are specialized filesystem structures designed to hold filesystem metadata. This metadata is not normally used by the filesystem. Instead, it is generally used for assigning specific security labels for kernel-level enforcement (e.g. Mandatory Access Control labels).

FAT12/16/32 (File Allocation Table 12-bit/16-bit/32-bit)

FAT12/16/32 is a DOS-based filesystem which is supported and used by 16-bit DOS (FAT12/16) and all versions of Windows (16-bit supported FAT12/16 while 32 and 64-bit versions support (FAT12/16/32) and 32-bit versions of DOS. These filesystems are natively supported by Linux.

Files-11

Files-11 is the native filesystem of the OpenVMS operating system and is considered as the progenitor of the NTFS filesystem. It is found on VAX, Alpha and Itanium OpenVMS-based systems.

Filesystem

A filesystem is a term used for a series of disk (and virtual disk) attributes and metadata (e.g. FAT, MFT, inodes, extents) which define how and where data is to be stored and distributed across a given storage device or partition.

FUSE (Filesystem in Userspace)

FUSE is a Linux kernel module which enables non-system administrative users to mount filesystems for which they normally would not have the authority. All FUSE code is run in userspace. Modern Linux systems fully support FUSE.

HFS (Hierarchical File System)

HFS is the native filesystem of the Mac OS operating system. This filesystem is natively supported by Linux.

HFS+ (Hierarchical File System/+)

HFS+ is the native filesystem of the Mac OS X operating system and supersedes HFS. This filesystem is natively supported by Linux.

HPFS (High Performance File System)

HPFS is a filesystem native to IBM's OS/2 operating system and supersedes the FAT filesystem in use at the time. This filesystem is natively supported by Linux.

JFS (Journaled File System)

JFS is a high performance journaled filesystem native to IBM's AIX UNIX platform. This filesystem is natively supported by Linux.

IDS (Intrusion Detection System).

An IDS system can be a hardware, software or hybrid system which scans a network, host system or both for signs of access violations or malicious activity based on various detection schemes including protocol scanning, trend analysis and base-lining differentiation and signatures. IDS systems generally report their findings to a management system or console and can be passive or active in nature.

Inode

An inode is a UNIX-based filesystem disk structure which stores information about a file or other filesystem object (its metadata) but does not actually store the file itself. However, inodes do contain the appropriate location or pointer to the file's actual physical disk location.

Linux

Linux is a free and open source software operating system based on UNIX. It was originally conceived and developed (the kernel) by Linus Torvalds, who holds and maintains the copyright to Linux. With the assistance of thousands of developers around the world, it has turned into a robust, stable and secure operating system, not unlike its commercial UNIX-based counterparts. Other than the kernel, it is mainly composed of developer-contributed software.

MFT (Master File Table)

The MFT is the basis by which the NTFS filesystem keeps track of all its files and associated metadata.

MINIX

MINIX is the native filesystem of the MINIX UNIX-like operating system developed by Andrew Tanenbaum. This filesystem is natively supported by Linux.

NIDS (Network Intrusion Detection System)

A NIDS is an information system used to detect network anomalies which are likely to indicate a network intrusion or network-related incident. These systems are available as both hardware and software solutions although most are a combination of both.

NTFS (New Technology File System)

NTFS is a high performance journaling filesystem native to Windows NT and all subsequent 32 and 64-bit Windows operating systems. This filesystem is natively supported by Linux. The NTFS filesystem largely resembles the OpenVMS ODS filesystem.

NTFS atime

NTFS (*atime*) is the date/time at which a file or directory was last accessed.

NTFS ctime

NTFS (*ctime*) is the date/time at which a file or directory was created or copied or moved to another filesystem. The act of copying or moving a file or directory to an altogether different filesystem causes that file or directory's *crtime* attribute to be set to the date/time of the copy or move.

NTFS crtime

NTFS (*crtime*) is the date/time at which a file or directory's MFT entries are modified, specifically when either a file or directory's MFT $DATA or $INDEX attribute is changed.

NTFS mtime

NTFS (*mtime*) is the date/time at which a file or directory is modified.

OpenVMS (Open Virtual Memory System)

OpenVMS is a high availability operating system designed to run on the DEC VAX, DEC Alpha and Itanium computer systems. OpenVMS is the descendant of VMS, the native operating system of the VAX.

Operating System

An operating system can be defined as a collection of binary executable code and system configurations separated into individual computer files and that controls the computer's hardware, operating system behaviour and the user-based applications, tools and utilities. An operating system is comprised of a kernel, a shell or GUI for interacting with the kernel and launching applications and user-based applications.

POSIX (Portable Operating System Interface for UNIX)

POSIX is a set of IEEE standards used for specifying API, shell and utility interfaces for maintaining a minimum level of compatibility between UNIX-like operating systems.

PTK

PTK is developed at DFLabs and is the premier open source GUI-based forensic suite. It uses The SleuthKit as its base investigative system but also provides numerous additional capabilities. A free version based on older PTK technology is available to the community for no charge but the more recent version with the latest capabilities is a for a fee product.

QNX

QNX is a commercially-based UNIX-like real-time operating system designed and targeted primarily at embedded device markets. QNX, in the context of this technical memorandum also refers to the QNX filesystem, referred to as QNX.

Reiser4

Reiser4 is the successor of ReiserFS and fixes many of its limitations. It is also a journaled filesystem. It is supported by Linux as a FUSE-based filesystem along with applied kernel patches.

ReiserFS (Reiser File System)

ReiserFS is a journaled all purpose filesystem developed by Hans Reiser. This filesystem is natively supported by Linux.

RFS (ReiserFS or Reiser File System)

See ReiserFS.

ROMFS (Read-Only Memory File System)

ROMFS is a very simple ROM filesystem whose primarily use is to imprint code onto integrated circuitry but can also be used by some systems to boot Linux kernels. This filesystem is natively supported by Linux.

Shell

See Command Shell

SysV (System V)

SysV is a generic term applied to various 1980's-era commercial System V UNIX-based filesystems. This filesystem is natively supported by Linux.

TCT (The Coroner's Toolkit)

TCT was written and developed by Dan Farmer, a former SGI employee and Wietse Venema, an IBM employee. It is considered to be the first forensic software toolkit for UNIX-based systems and is the predecessor of The SleuthKit.

The SleuthKit

The SleuthKit, based on TCT, is the premier open source forensic software investigative suite upon which other successful frameworks, including PTK and Autopsy, use as their basis. The SleuthKit is developed primarily by Brian Carrier and is under ongoing development.

TLN (TimeLiNe)

The TLN file format is an intermediate timeline format which is used as a mid-point between a filesystem's raw date/time metadata structures and a readable text-based timeline. It was proposed by Harlan Carvey and now presents a viable alternative to the Bodyfile format. However, it is far less complete than the Bodyfile intermediate format.

UDF (Universal Disk Format)

UDF is the native DVD filesystem and was designed to supersede CDFS. It suffers fewer limitations than its predecessor (CDFS). This filesystem is natively supported by Linux.

UFS (UNIX File System)

UFS is a generic term meant to denote multiple similar filesystems including FFS, UFS1 and UFS2 found on various UNIX and BSD operating system systems including HP and SUN UNIX servers. Some versions of UFS support filesystem journaling. This filesystem is natively supported by Linux.

UNIX

UNIX is a multi-user, multitasking, multithreaded operating system based on a kernel that provides a consistent interface to the user for both interactive and background job processing. UNIX is multi-platform and hardware independent and it supports advanced APIs and system calls. It is generally considered by the computing industry as the hallmark of scalable, robust, secure and reliable computing. It was originally developed at AT&T Labs by Dennis Ritchie and Ken Thompson, who were developing a more programmer-friendly operating system.

UMSDOS

UMSDOS is a native DOS-based Linux filesystem which allows both DOS and Linux to co-exist on the same partition. This filesystem is natively supported by Linux.

UTC (Universal Time, Coordinated)

Universal Time, Coordinated or Coordinated Universal Time as it is more commonly referred to is the international time standard. It is the most commonly used term for what once use to be referred to as Greenwich Mean Time. Universal Time runs on a 24-hours clock where zero (0) hours passes through Greenwich and runs eastward in time.

VxFS (Veritas File System)

VxFS is an advanced commercial UNIX-based filesystem supported on multiple platforms and architectures. It is available on commercial-only basis for Linux.

Windows

Windows is a graphically-based operating system developed by Microsoft. It was first introduced in 1985 and since that time, has undergone many changes and iterations. Now, it is a multithreaded, multiprocessor, multitasking operating system and is also the most popular desktop operating system currently in use worldwide. Due to its prevalence, it is also the most heavily attacked operating system worldwide. Because the operating system must cater to a very large variety of computing platforms and hardware peripherals, it lacks certain security features found in more hardware-specific operating system implementations. However, great strides have been made over the recent years to improve its security. Past iterations have included Windows 3.x, 95, 98, NT, ME, 2000, XP, 2003 and Vista.

XENIX

XENIX is a native filesystem to the XENIX UNIX-based operating system of the late 1970s. This filesystem is natively supported by Linux.

XFS (eXtended File System)

XFS is a high performance journaled file system native to SGI UNIX workstations, servers and supercomputers. This filesystem is natively supported by Linux.

Xiafs

Xiafs is a native Linux filesystem developed by Frank Xia based largely on the MINIX filesystem. This filesystem is natively supported by Linux.

ZFS (Zettabyte File System)

ZFS is a highly advanced UNIX filesystem designed primarily for data redundancy and reliability on multi-disk devices including RAID arrays and provides allocable storage pools. It is supported by Linux as a FUSE-based filesystem. Rather than rely on extents and inodes it is based on pointer block allocation.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

UNCLASSIFIED
(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC JUNE 2010

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Generating computer forensic super-timelines under Linux: A comprehensive guide for Windows-based disk images

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Carbone, R.; Bean, C.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
October 2011	132	84

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
31XF20 « MOU RCMP "Live Forensics" »	

10a. ORIGINATOR's DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Valcartier TM 2011-216	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

(U) This technical memorandum examines the basics surrounding computer forensic filesystem timelines and provides an enhanced approach to generating superior timelines for improved filesystem analysis and contextual awareness. Timelines are improved by polling multiple sources of information across the filesystem resulting in an approach that is surprisingly flexible and customizable. The timeline is further enhanced by incorporating key time-based metadata found across a disk image which, when taken as a whole, increases the forensic investigator's understanding.

(U) Ce mémorandum technique examine les bases entourant la création d'un calendrier des événements inforensiques des systèmes de fichier et fournit une approche améliorée pour générer des calendriers supérieurs pour une analyse améliorée des systèmes de fichiers et un meilleur éveil contextuel. Ces calendriers sont améliorés en sondant des sources multiples d'information à travers le système de fichiers, ce qui résulte en une approche qui est étonnamment flexible et configurable. Le calendrier est amélioré encore davantage par l'introduction des métadonnées essentielles liées au temps qui se retrouvent un peu partout sur un disque et qui, lorsque prises en compte globalement, augmentent la compréhension de l'enquêteur inforensique.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Bodyfile; Computer Forensics; Digital Forensics; File attributes; Forensic investigation; Log2timeline; Mactime; Super-Timelines; The Sleuth Kit; Timeline analysis; Timeline formats; TLN format